SOFTWARE RE-ENGINEERING

IN EUROPE

INPUT

About INPUT

INPUT provides planning information, analysis, and recommendations for the information technology industries. Through market research, technology forecasting, and competitive analysis, INPUT supports client management in making informed decisions.

Subscription services, proprietary research/consulting, merger/acquisition assistance, and multiclient studies are provided to users and vendors of information systems and services. INPUT specializes in the software and services industry which includes software products, systems operations, processing services, network services, systems integration, professional services, turnkey systems, and customer services. Particular areas of expertise include CASE analysis, information systems planning, and outsourcing.

Many of INPUT's professional staff members have more than 20 years' experience in their areas of specialization. Most have held senior management positions in operations, marketing, or planning. This expertise enables INPUT to supply practical solutions to complex business problems.

Formed as a privately held corporation in 1974, INPUT has become a leading international research and consulting firm. Clients include more than 100 of the world's largest and most technically advanced companies.

INPUT OFFICES

North America

San Francisco
1280 Villa Street
Mountain View, CA 94041-1194
Tel. (415) 961-3300 Fax (415) 961-3966

New York
Atrium at Glenpointe
400 Frank W. Burr Blvd.
Teaneck, NJ 07666
Tel. (201) 801-0050 Fax (201) 801-0441

Washington, D.C.
INPUT, INC.
1953 Gallows Road, Suite 560
Vienna, VA 22182
Tel. (703) 847-6870 Fax (703) 847-6872

International

London
INPUT LTD.
Piccadilly House
33/37 Regent Street
London SW1Y 4NF, England
Tel. (071) 493-9335 Fax (071) 629-0179

Paris
INPUT SARL
24, avenue du Recteur Poincaré
75016 Paris, France
Tel. (1) 46 47 65 65 Fax (1) 46 47 69 50

Frankfurt
INPUT LTD.
Sudetenstrasse 9
W-6306 Langgöns-Niederkleen, Germany
Tel. 0 6447-7229 Fax 0 6447-7327

Tokyo
INPUT KK
Saida Building, 4-6
Kanda Sakuma-cho, Chiyoda-ku
Tokyo 101, Japan
Tel. (03) 3864-0531 Fax (03) 3864-4114

MAY 1992

SOFTWARE RE-ENGINEERING IN EUROPE

INPUT

Piccadilly House, 33/37 Regent Street, London SW1Y 4NF, U.K. +44 71 493 9335
24, avenue de Recteur Poincaré, 75016 Paris, France +33 1 46 47 65 65
Sudetenstrasse 9, D-6306 Langgöns-Niederkleen, Germany +49 6447 7229

Published by
INPUT
Piccadilly House
33/37 Regent Street
London SW1Y 4NF
United Kingdom

Information Services Programme - Europe 1992

Software Re-engineering in Europe

IERE2 · 567 · 1992

Abstract

This report analyses the use of CASE tools and methodologies in applications development by IS departments and vendors. Software and services vendors have traditionally concentrated on new application projects. INPUT's research identifies the size and growth in the use of these tools and the likely timescales in which software re-engineering will become a common practice. Re-engineering existing applications within a modern software engineering environment may become the major opportunity for overloaded IS departments and vendors of CASE products and services in the 1990's.

The report examines the pressures on IS departments to adopt new working practices, and the timescales in which they expect to see a reduction in software maintenance costs. Payback, organisational impact, buying priorities and the response to client-server architectures are all investigated.

User research covers France, Germany and the U.K. - the largest markets for CASE tools in Europe. It provides insights into the success or otherwise of these tools in both large and small organisations.

The report provides an assessment of the state of the re-engineering market. Its will be of particular value not only to vendors but also to IS departments with heavy application maintenance workloads. It looks to test the validity of the promise that re-engineering will be a cost effective way of maximising the return on existing IT investments.

Table of Contents

Table of Contents (Continued)

Exhibits

Exhibits (Continued)

I
Introduction

A
Scope and Objectives

This report has been produced as part of INPUT's European Information Services Programme which analyses the development of the computer software and services industry.

The main focus of the report is on two issues impacting the emergence of re-engineering:

- The experience and expectations of IS departments in the use of CASE and related initiatives.
- The underlying architectural advances which are impacting demand for application development environments.

INPUT's forecast is for the size of the CASE product market. It is currently not feasible to measure the separate impact of CASE on other areas such as professional services.

The objective of the report is to assess the impact of CASE tools and methodologies on the demand for re-engineering of existing software applications. The assessment includes:

- an estimate of the market size and growth of CASE products in the major geographic regions.
- a detailed analysis of user experience to date and opinions on outstanding issues.

- a commentary on the major market trends to be expected in the area of re-engineering.

B
Definitions

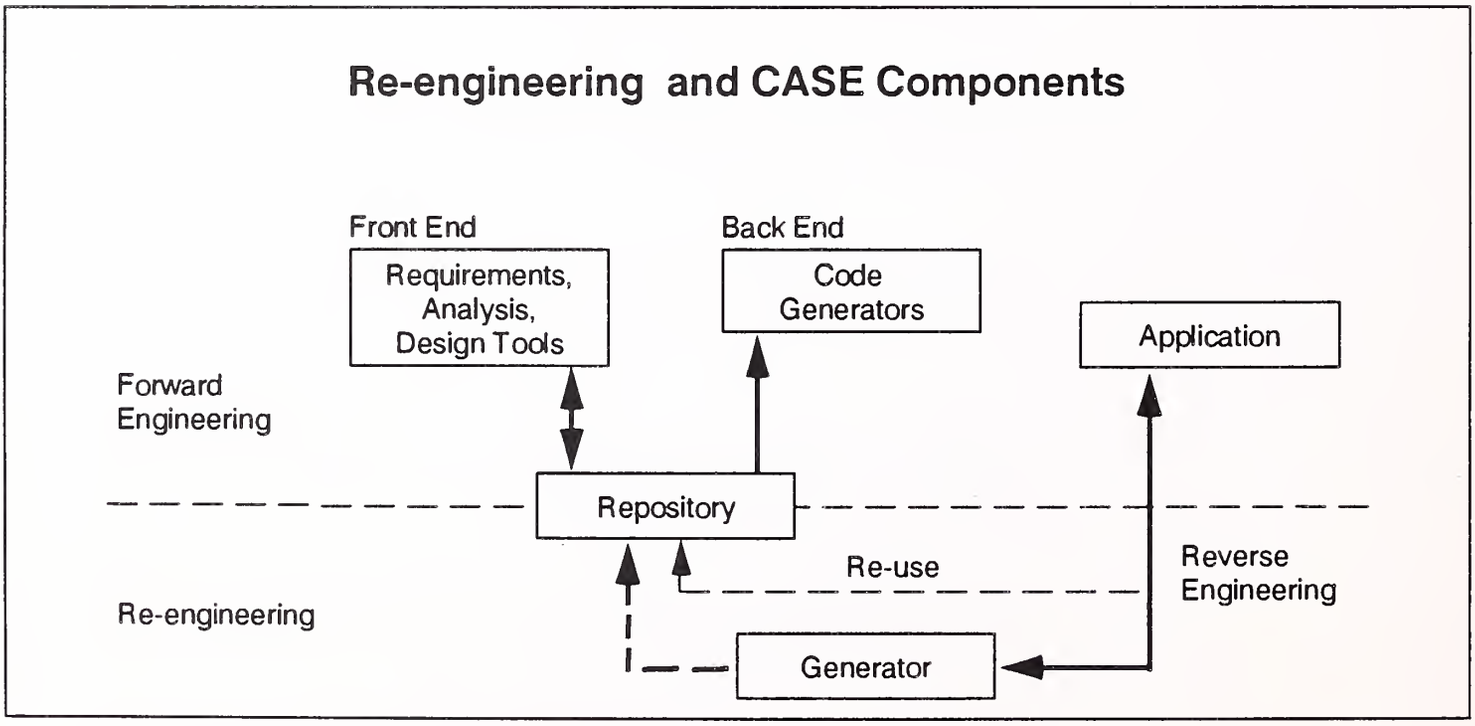
Software Re-engineering and CASE (Computer Aided Systems Engineering) are terms which are applied rather liberally within the software and services industry. For the purposes of this report the following definitions are used:

All references to CASE are in the context of tools to support the business applications development processes. CASE, in this context, includes:

- Forward engineering
- Re-engineering
- Repository technology

Exhibit I-1 lays out the major components of CASE and their relationships.

Exhibit I-1



Forward engineering has traditionally been divided into:

- Front-end (or Upper CASE) tools for performing business modelling, requirements analysis and design work
- Back-end (or Lower CASE) tools, or code generators.

In the 1980's front-end and back-end tools were generally separate. With the advent of repository technology, this has begun to change quickly.

- A repository (or encyclopedia, or dictionary) is used to capture requirements and designs on a centralised or shared basis.

- The repository can maintain changes and serve as the start point for code generation.

In this report re-engineering is used to describe the entire process of taking an existing software application and *either* re-using the logic *or* reverse-engineering the code.

The term "business re-engineering" implies a totally different process - the re-structuring of a business in order to become more competitive and exploit information systems investment more fully.

C
Methodology

The following sources were used for this report:

- Telephone and face-to-face interviews with 64 senior IS executives with responsibility for application development. Appendix A gives an analysis of their business sector, size of IS function and country.

- Telephone and face-to face interviews with leading CASE and systems vendors in both Europe and the U.S.A..

- INPUT's library and databases of the European Software and Services Industry, and related studies.

D
Report Structure

Chapter II is a summary of the entire report aimed at highlighting the key issues from the research and INPUT's recommendations.

Chapter III describes the development of CASE, its contribution to improving application development environments and the role of re-engineering.

Chapter IV analyses the results of detailed CASE user research, identifying their experience to date and their views of the future for re-engineering.

Chapter V provides INPUT's market forecast for CASE products in Europe over a five year time span.

Chapter VI identifies key conclusions and recommendations for users and vendors actively assessing the software re-engineering process.

E
Related INPUT Reports

Please refer to the following related INPUT reports:

Operational Software Support and Maintenance, Europe, 1991

The Future of CASE, 1991-1996 - U.S.A.

Systems Software Products, 1991-1996 - Europe

Professional Services, 1991-1996 - Europe

II
Executive Summary

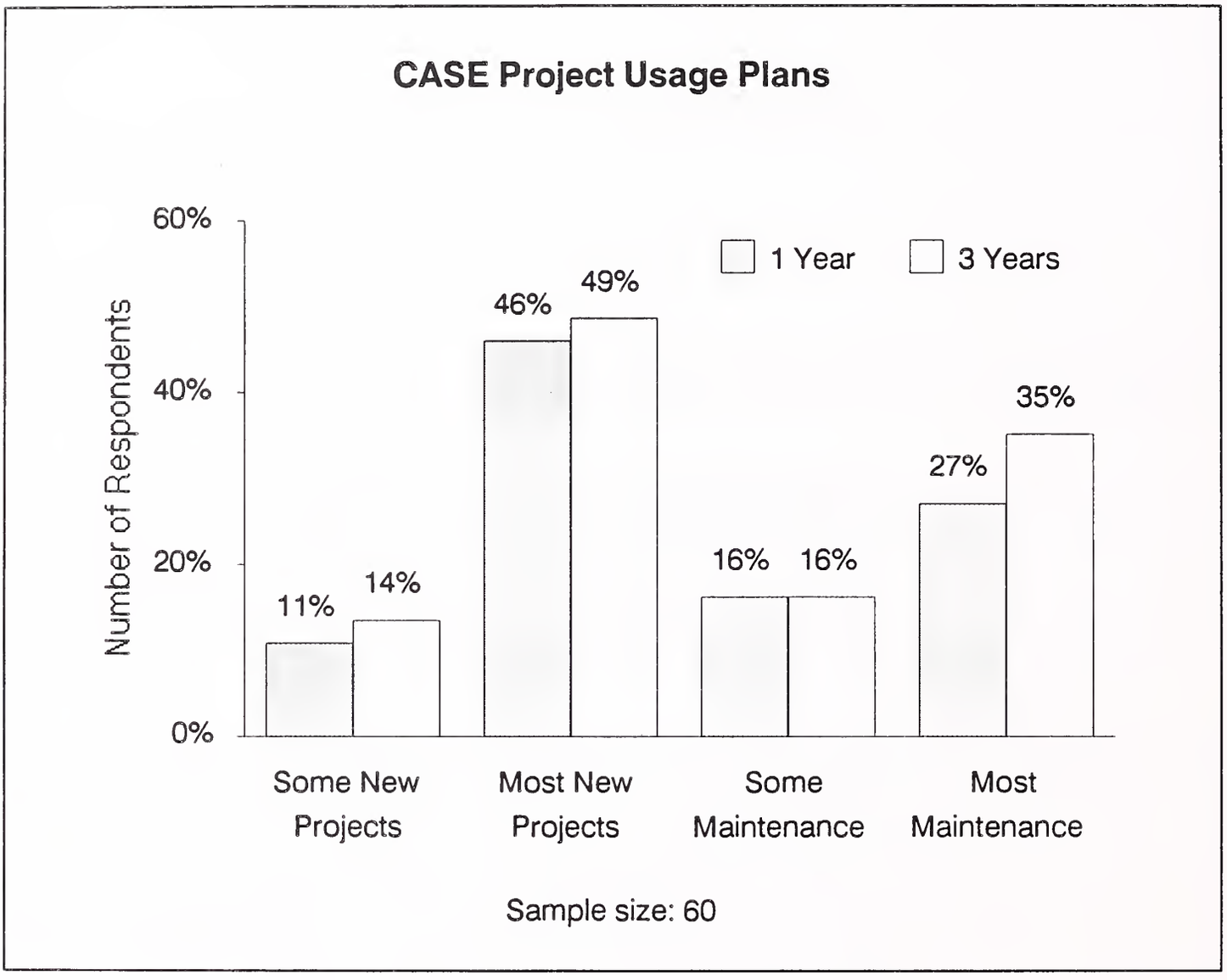
A
Software Maintenance Benefits from CASE

As competitive pressures mount between software and service vendors, more and more are turning their attention toward the opportunity represented by their clients' ponderous maintenance workload. The ability to re-engineer existing applications, to migrate them onto new platforms and to manage them within a CASE-based development environment is turning from a nice theory to a practical reality.

INPUT research has identified that CASE is already starting to reduce maintenance costs and problems for over 40% of its users. In February and March 1992, sixty IS managers with responsibility for the application development environment were interviewed in France, Germany and the United Kingdom. Between them their organisations had spent over $30 million on CASE products to date and expected to spend nearly $40 million more over the next two years.

Exhibit II-1 shows how current users of CASE tools will be using CASE to support their development and maintenance teams in 1992 (within one year) and their expectations for 1995 (in three years time).

EXHIBIT II-1



Several respondents are still proceeding CASE tools cautiously - about 20% of the sample were not yet convinced that they would be adopting CASE for anything other than a limited trial. Nearly one third were also sceptical that they would ever find the use of CASE reducing their maintenance costs. However the large majority are adopting CASE strategies wholeheartedly, along with a broad range of other initiatives aimed primarily at improving application quality and project team productivity. Some of these other initiatives are listed in Exhibit II-8.

In recent years, with the software and services industry growing between 20% and 30% per year, few vendors were paying much attention to their clients' software maintenance workload - typically consuming 65% of in-house development resouce. The focus was all on new applications.

In the past two years this has changed as competitive pressure has increased and open systems have gained commercial credibility. The money spent internally by IS departments on maintaining and enhancing their applications is now seen as an attractive opportunity for vendors. Added attractions are the concepts of re-engineering and migration - bringing old applications forward into new environments and onto new platforms - as ways of keeping clients loyal to their incumbent supplier.

Many major system vendors are now packaging up attractive software CASE tools and services to encourage clients to carry forward existing applications investment onto new platforms - from the same vendor. In the world of open systems, migration to other vendors' platforms has become a lot easier. Downsizing or rightsizing re-engineered applications can be an attractive option compared to starting again from scratch.

B

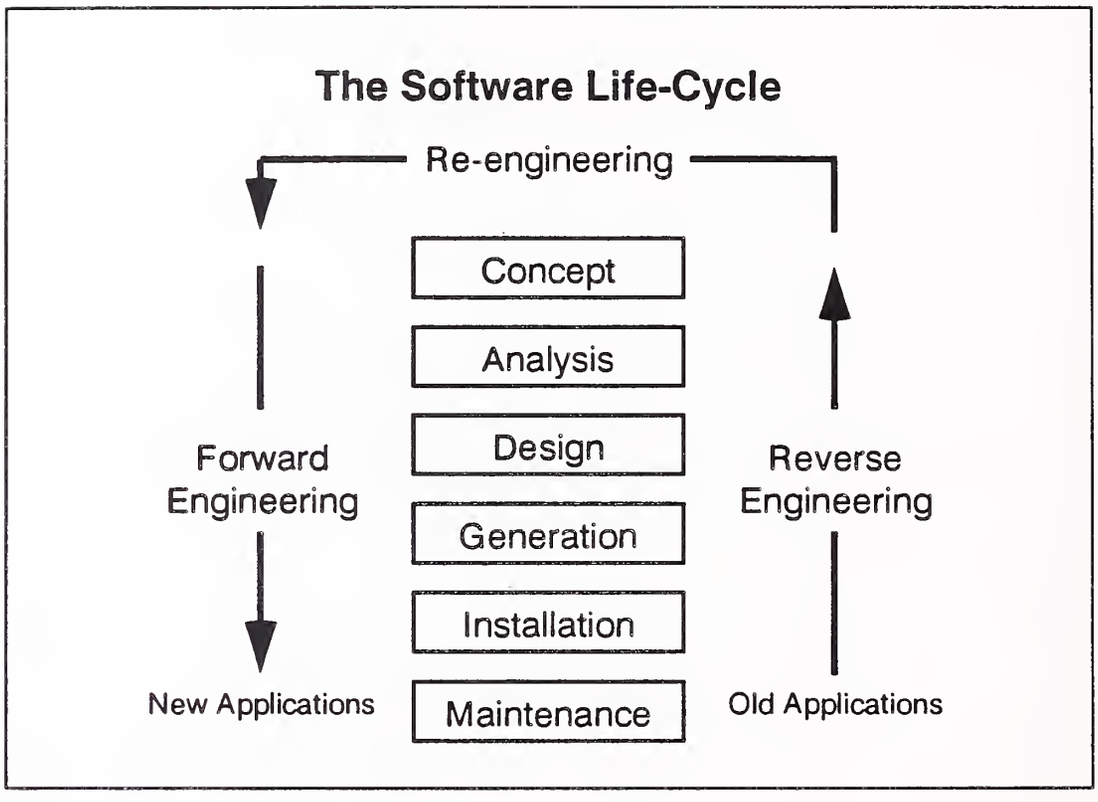
Application Development Pressures

The pace of change in business has never been greater, stimulated to a major extent by the availability of more timely and relevant information throughout the business process. As if this wasn't enough to keep the pressure on application development and maintenance teams, new innovative software and hardware technology also continually changes the ground rules. How to keep pace with business demands and technology's cost benefits is the challenge to which the CASE industry is responding.

Exhibit II-2 is one simple way of viewing the stages that a software project goes through from original concept to life-extending maintenance. Depending on the state of the software it can be considered to be following one of three essential processes:

- Forward engineering
- Reverse engineering
- Re-engineering.

EXHIBIT II-2



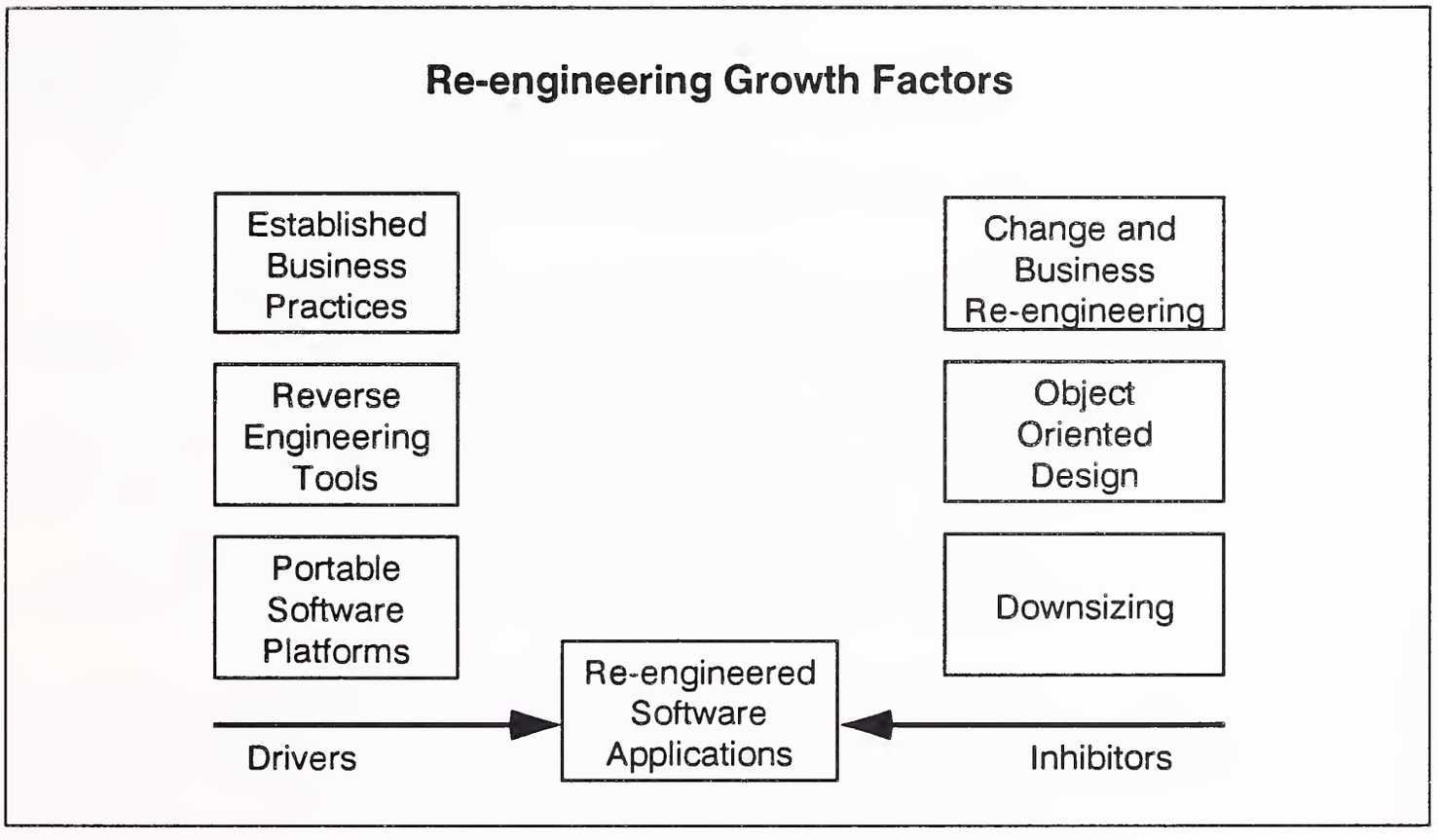
Forward engineering has been the focus of the CASE industry for both tools and methodologies. More efficient and effective ways of tackling the development of new applications has been the goal.

Reverse engineering is a more recent phenomenon, catering for re-generating application documentation from heavily modified or un-documented code.

Re-engineering is the third phase, applying modern engineering principles to the updating of ageing applications.

The factors influencing growth in software re-engineering are shown in Exhibit II-3.

EXHIBIT II-3



The feasibility and growth of re-engineering depends heavily on:

- A clear recognition that the value of existing applications lies in their proven structure, the implied user working practices and users' familiarity.
- The availability and adoption of life-cycle CASE tools for reverse engineering poorly documented applications, updating their functionality, and then forward engineering them for re-use on modern software platforms.
- The development of methodologies which properly account for a pragmatic re-engineering process, rather than assuming that an enterprise business model already exists or that applications are best given a fresh start from a clean sheet of paper.

Re-engineering is an attractive process for software or system vendors with large customer bases. The advent of open systems and downsizing continually threatens to lose them customers to their competitors. Re-engineering in the form of cost effective migration tools and procedures can minimise this threat, keep customers loyal, and allow them to move to new software platforms when appropriate. These vendors have a vested interest in helping their clients carry forward their past software investments.

The trend to downsizing and open systems is encouraging the development or purchase of re-usable, scalable applications. Only a few of these are going to be based on existing re-engineered designs.

Tools and techniques for developing applications are still improving rapidly. Most users have their eyes on Object Oriented software, even though it may be several years before it will be used widely. It is unlikely that applications designed without object oriented program systems (OOPS) in mind can be re-engineered to that architecture.

There is a growing realisation that many IT projects have resulted in businesses "automating their problems". Re-addressing the way they do business and re-engineering the business to make best use of IT are attractive radical options. Few old software applications are likely to survive for long and fewer perhaps will get re-engineered to fit such new regimes.

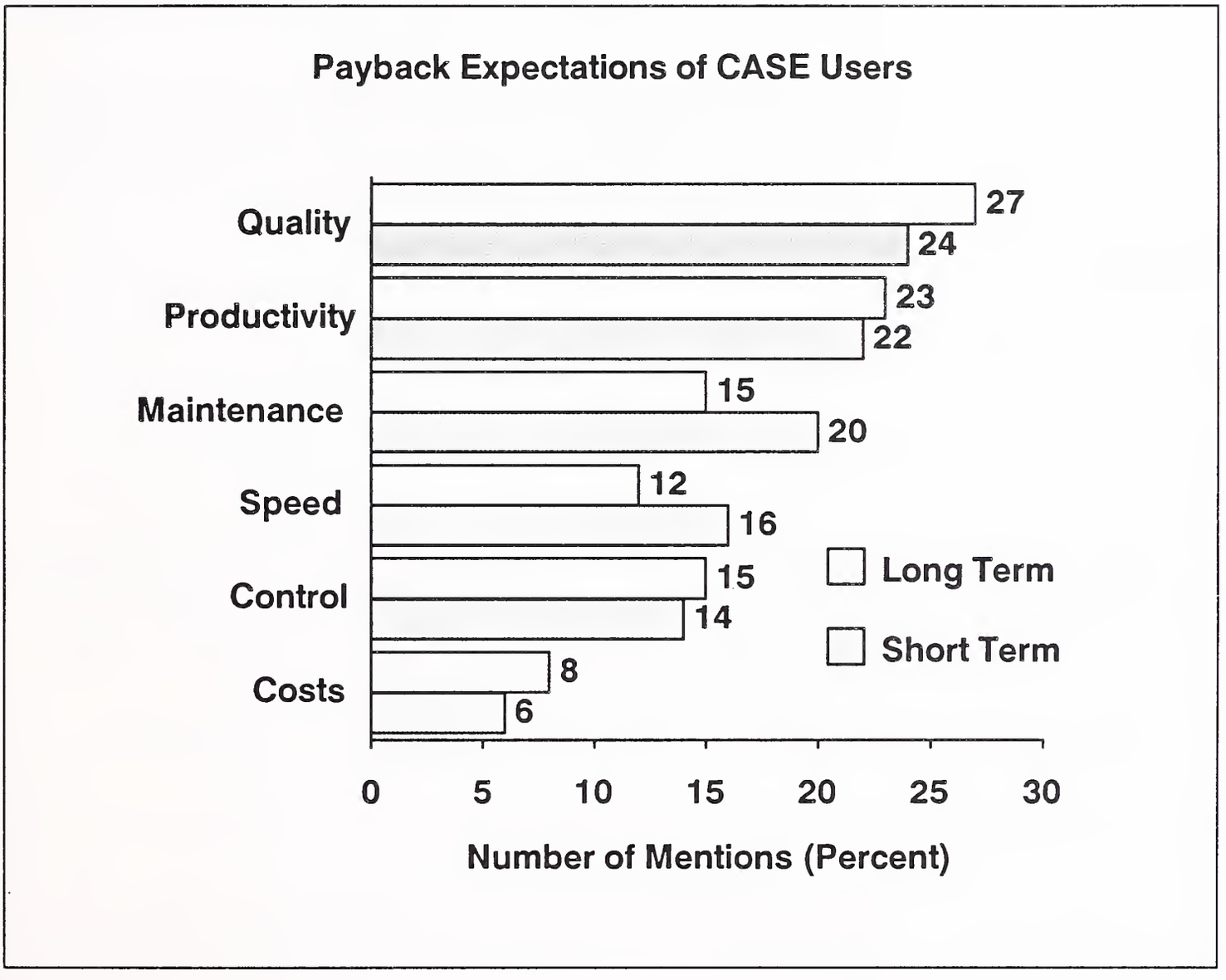
C
User Requirements, Issues, Trends

In the survey upon which this report is based, an extraordinary variety of answers were given to the question "What do you expect to be the principal short and long term payoffs from CASE?". There is no single over-riding result driving these complex decisions. For the purposes of analysis the answers were categorised into six primary concerns:

- Quality improvements of all types
- Productivity gains for developers
- Improvements to the maintenance process
- Speedier response to user needs
- Better management control
- Reduction in costs.

In Exhibit II-4 quality, productivity and maintenance are the areas in which respondents expected the most benefit. Cost reduction was expected in the short term by only a handful of respondents.

EXHIBIT II-4



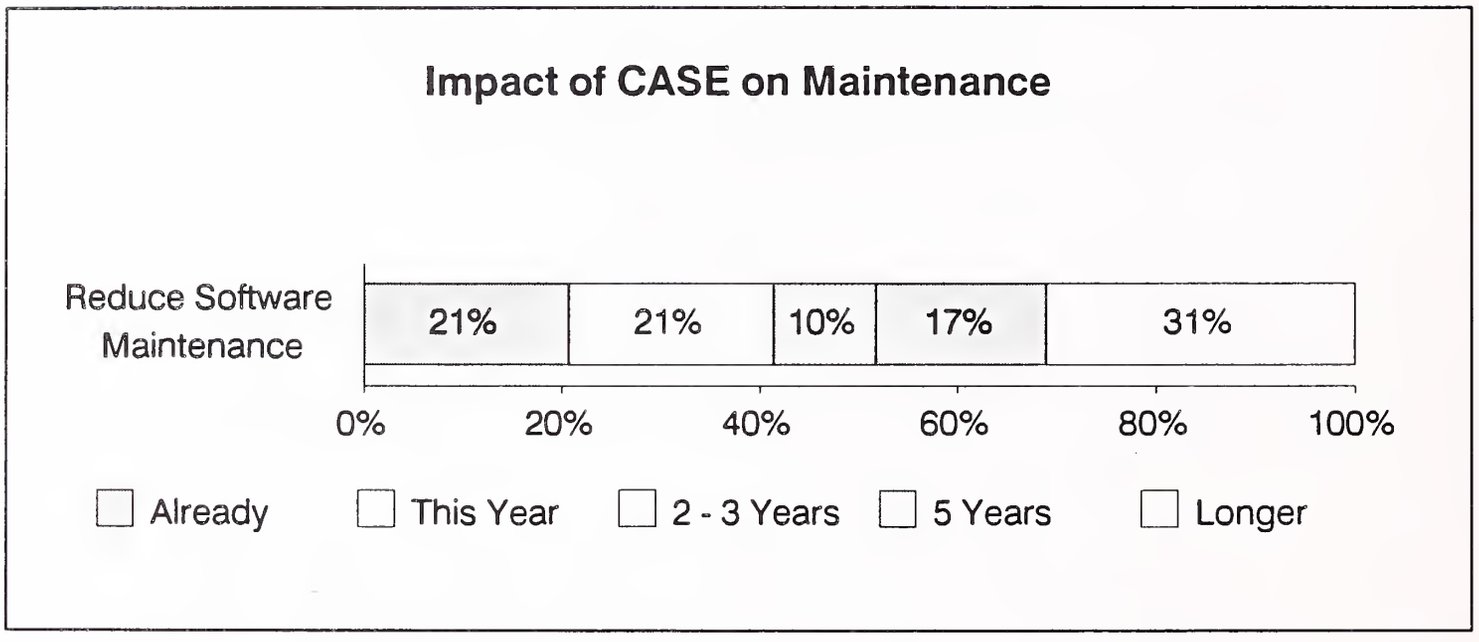
In the long term the expectations change. Quality improvements take a clear lead, but maintenance expectations fall.

There are clear indications that many respondents were applying CASE to their documentation problems, using the tools to establish consistent documentation of existing software. This, in turn, is expected to give early improvement to their software maintenance workload. In the longer term maintenance is then much less of an issue than productivity and quality across the whole application development environment.

Re-engineering is the term applied to carrying forward an existing investment in application software to new hardware, software and development environments. The promise of re-engineering is that it will reduce the costs of software maintenance and improve responsiveness to changing user needs. When will it start to fulfil this promise?

To some extent it already has started. Exhibit II-5 shows the spread of responses to the question "When do you expect your use of CASE to begin to reduce your software maintenance costs?" Over 40% say it will have started making an impact during 1992.

EXHIBIT II-5



However the Exhibit also shows that a significant one third of respondents are very sceptical that they will ever reduce maintenance costs.

This parallels the results of research INPUT did last year into software maintenance, where the main obstacle to improvement was found to be the lack of awareness among IS management that there were improvements to be made. For example there was little awareness of the possibility of outsourcing maintenance, the use of reverse engineering or re-engineering CASE tools, or the application of maintenance methodologies and working practices.

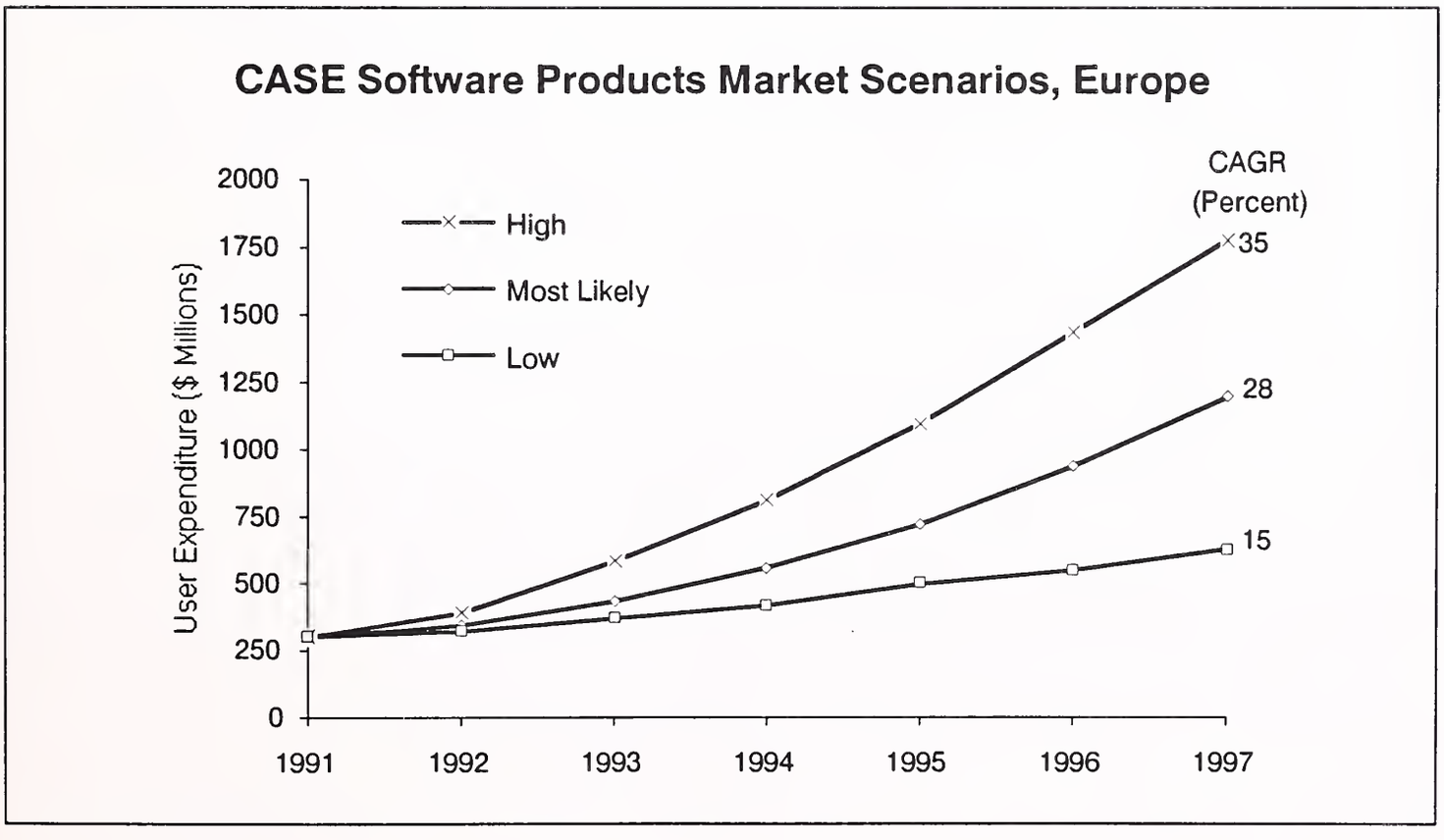
INPUT's 1991 research suggested that the crux of the software maintenance problems was a lack of measurement and management processes within IS departments.

D
CASE Growth

Users surveyed expected, on average, to spend 32% more on CASE in the next two years than they had spent to date - an expectation which runs counter to recent press reports of CASE failing to deliver. INPUT's forecast takes a more conservative view. The most likely growth scenario for CASE products in Europe in the 1990's is indicated in Exhibit II-6:

- This represents an increase from about $350 million in 1992 to $1.2 billion by 1997, a compound annual growth rate of 28%.
- INPUT expects annual growth in 1992 to be 15%, peaking around 1995 at 30%. The pattern is likely to be different for the major country markets as shown in Exhibit II-7.

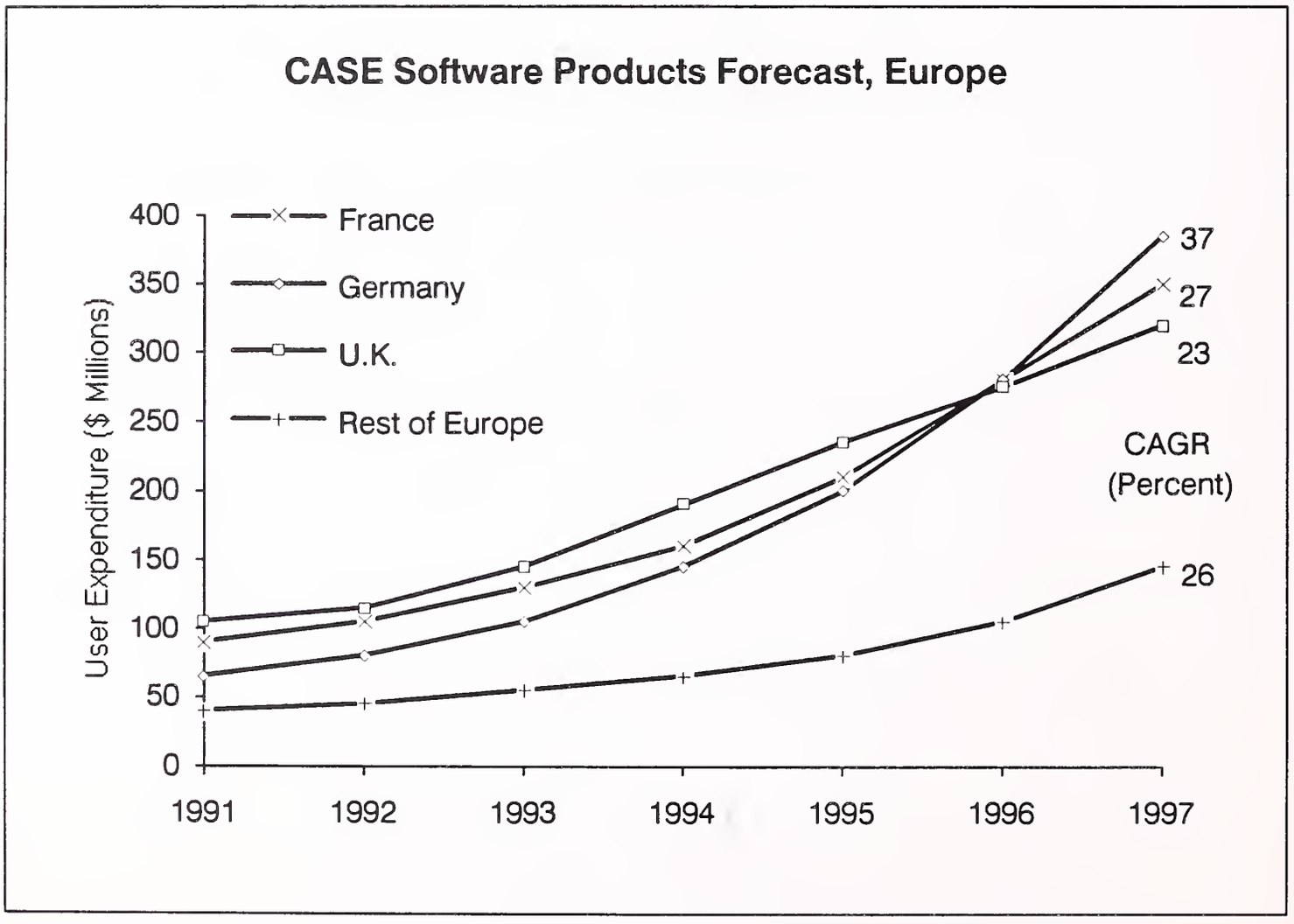
EXHIBIT II-6



The market shown here is for CASE products only, and excludes CASE-related services which at present are far more difficult to quantify.
The "High" and "Low" scenarios each have a 25% probability of occuring, compared to a 50% probability for the "Most Likely" growth.

The most likely growth for each of the major countries is shown in Exhibit II-7. Each is expected to follow a significantly different pattern of growth over the five years. Each market is at a different stage of development and subject to its own economic pressures.

EXHIBIT II-7



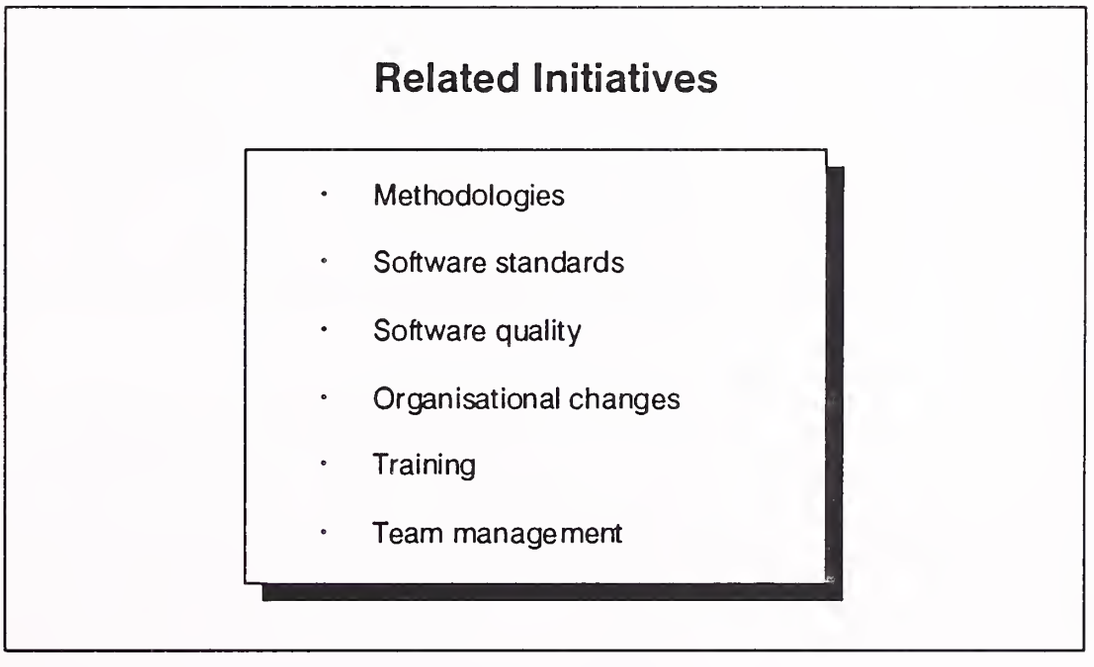
Growth will be significantly impacted by two distinct variables:

- Organisational issues concerning the readiness of application development teams to absorb and use CASE tools and technology.
- The extent to which re-engineering technology fulfills its promises and is put to use.

The first of these - organisational readiness - is a particular opportunity for IS and management consultancy vendors. The second - re-engineering technology and techniques is an opportunity for product, methodology and training vendors.

Decisions on CASE can rarely be treated as independent of other issues. CASE products and skills are merely one element of a more complex application development environment. The most important issues, as perceived by respondents, are listed in Exhibit II-8.

EXHIBIT II-8



Most closely related to CASE are the methodologies used for system development and maintenance. These vary significantly from country to country and from organisation to organisation. Some respondents have developed their own methods and working practices, others have adopted the national standards such as Merise in France or SSADM in the United Kingdom.

Many interviewees pointed out how difficult it is to introduce a methodology after having made a major CASE purchase. Methodologies usually offer some freedom in choice of CASE product, but frequently the reverse is not true. Many CASE products cannot be integrated into a variety of methods.

Adoption of standard software platforms or techniques can also restrict the choice of CASE products. After considering over 300 potential products, one respondent found only a handful of products which would support their strategy for co-operative processing.

Adoption of new tools and methods cannot usually be done effectively without some re-organisation in working practices and reporting structures. Just as a business must re-consider its business processes in order to exploit IT fully in the market, so an IS department must re-consider its application development processes in order to exploit new technologies and serve its business clients more efficiently.

The advent of CASE-based application development has raised a new issue in the area of training. Software development staff are not only being asked to change their working practices, they are being asked to extend their skills and become fully qualified systems or software engineers. This is a daunting task for them and their management. A large element of their current skill has been learned on the job within an IS department, often in a rather ad hoc way. Turning professional staff into information engineers requires a major commitment to training.

In 1992 INPUT expects little maintenance, modification, redevelopment and new development to be performed using re-engineering tools. This low usage is due to a lack of critical mass in re-engineering, and the slow response of user management to the ideas underlying re-engineering concepts. Even with several major vendors now marketing re-engineering and maintenance tools and services there is still a marked reluctance among their clients to take firm decisions on re-engineering their critical applications software.

By 1997 INPUT expects this picture to have turned around markedly - essentially because all (or most) of the factors above will have been reversed.

E
Recommendations

IS managers need to carefully assess the choices open to them in their applications investment. The choices to be made for each existing application can be summarised as:

- Drop - the benefit no longer justifies cost
- Hold - minimise support and maintenance
- Carry forward - improve service to users
- Re-new - replace with new system by:
  - Re-engineering and migrating
  - Developing new custom application
  - Buying suitable application package

Re-engineering offers software and services vendors a set of opportunities which they have been able largely to ignore in times of rapid IS market growth.

There are several types of vendor active in this market:

- The I-CASE vendor.
- The product vendor with powerful forward engineering tools.
- The specialist reverse engineering CASE vendors.
- The methodology vendors.
- The professional services vendors.

For the CASE vendor there are two main challenges emerging as re-engineering becomes a financially viable option for users:

- How to package up the tools which would allow their user base of clients to reconstruct the specification and documentation of existing applications. This form of reverse engineering option can offer an early payback to the client by merely improving his ability to manage the maintenance workload.

- How to then re-engineer and migrate the most valued of these old applications into the environment used for new application development? Payback for the client is likely to be much longer term than the reverse engineering phase. Those vendors who primarily sell a software platform can be expected to have an advantage over the purely CASE product vendor.

The I-CASE vendors will need to extend their field of influence in order to support users with an ever more complex mix of tools and applications. There is a strong need for resource management tools, to allow IS management a fuller view of the projects and staff under their management. The complexity of this task is not expected to ease as a result of downsizing or distributed systems.

In general, CASE vendors will probably need a wider variety of business partners in order to address the re-engineering market opportunity. It will be essential to be able to understand the financial implications of all the different choices each client faces in determining the most effective way to engineer an application - see section B "Applications Development Pressures" above.

Specialist or niche vendors need a set of established partners to meet users need for all complex aspects of the re-engineering process. These include having the skills to:

- Value existing applications within a business.
- Cost alternative approaches to up-dating applications (re-engineer, re-invent, buy-in, etc.).
- Supply reverse engineering tools.
- Integrate the new documentation with other enterprise models using latest methodologies.
- Support the new life-cycle process.

These skills range from those traditionally considered management consultancy through to detailed platform software. Most product vendors will partner a range of professional service organisations.

For the professional services vendor the main opportunity is in using these tools and methods to re-engineer applications for their clients. Over the next five years this could well begin to contribute more than 50% of their application development revenues.

There is no doubt that, given the right tools, re-engineering will offer IS departments the best payback on their huge applications investments.

III
Application Development Environments

Systems engineering, and its manifestation as CASE, can sometimes appear to be an end in itself. Academics and researchers (as well as self-interested vendors) discuss at length their positions on such issues as methodologies, data and process representation, the place for object-oriented design, information modeling alternatives, etc. These are all important issues. However, the ultimate justification for CASE is how useful it is to application developers and their management.

This chapter will examine issues that relate to the evolution of application development environments:

- Software life-cycles
- The maintenance workload
- Emerging architectures
- Forward engineering
- Reverse engineering
- Re-engineering.

A

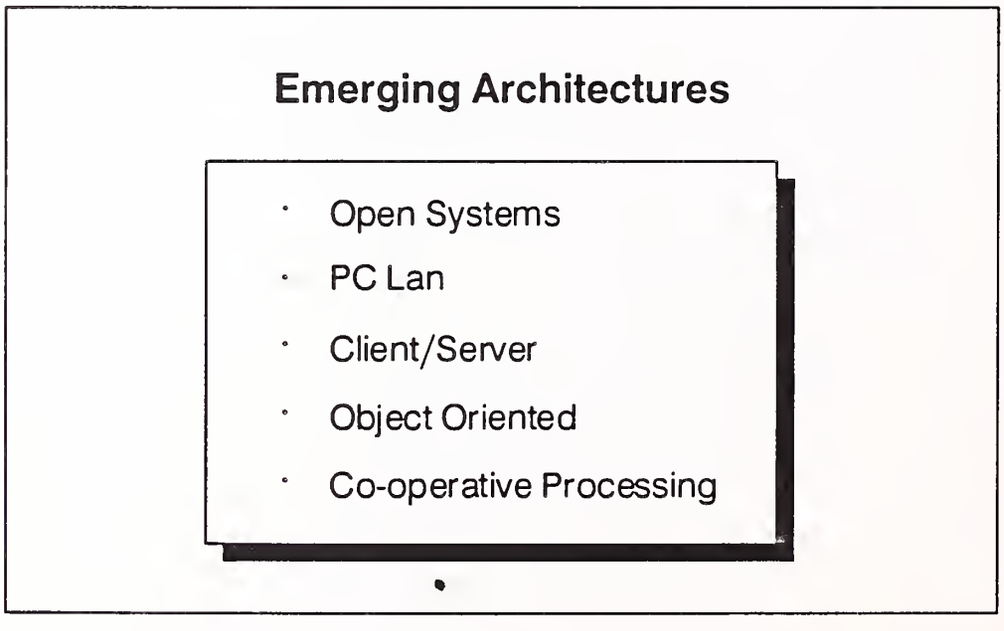
Architectural Changes

Most IS managements are today trying to set down standards with which to establish a "better application development environment". Their horizon keeps moving as the organization they serve changes its business objectives and processes. They face further changes as new technology emerges promising lower costs, higher quality and flexibility.

Aggravating the problems of deciding on new working practices are the emerging software and hardware architectures, listed in Exhibit III-1.

The large complex systems which have been produced in the past have been mainframe or large minicomputer based. Today the downsizing phenomenon is changing all that and powerful low-cost systems are being networked together (or left to operate autonomously) resulting in many applications being off-loaded from their traditional mainframe environment.

EXHIBIT III-1



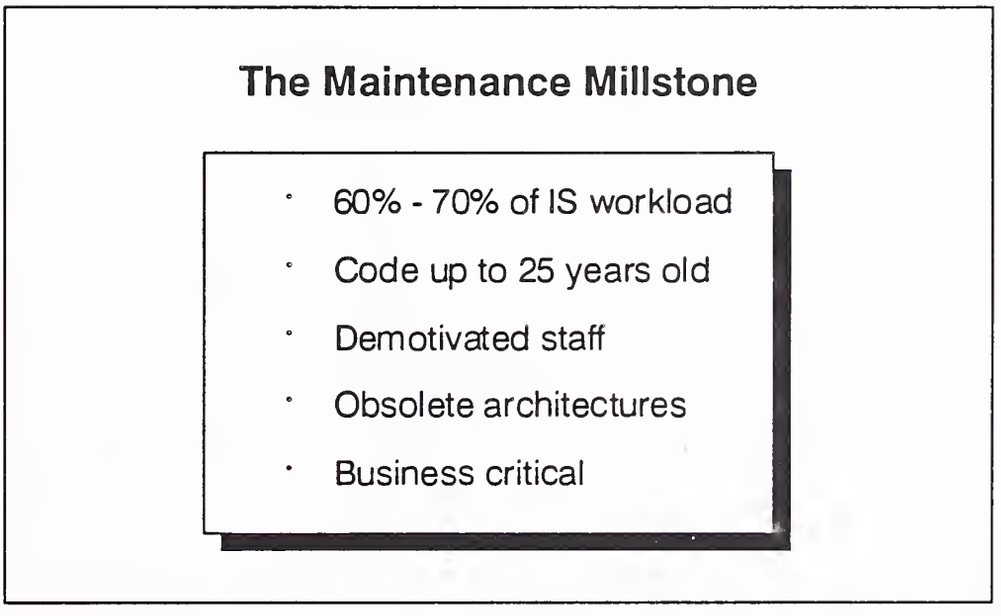
Each of these architectural changes promises to lower costs, but most of them represent new obstacles in terms of carrying forward existing investments in software.

Software is expensive. In many businesses software that survives for any period of time greater than two or three years becomes the embodiment of the business itself. The system is the business and the dependence of the business on that software increases its life and adds to the investment in it. Such software is often called business-critical.

For example there are core applications running in some banks and insurance companies which are over 25 years old. Such old software is both an asset - it may be absolutely essential to certain operations - and a huge liability - requiring excessive use of skilled resources to maintain it in tune with changing needs.

The millstone of software maintenance is a major problem for most IS departments, consuming the majority of staff and holding back the adoption of new skills and technologies - see Exhibit III-2.

EXHIBIT III-2



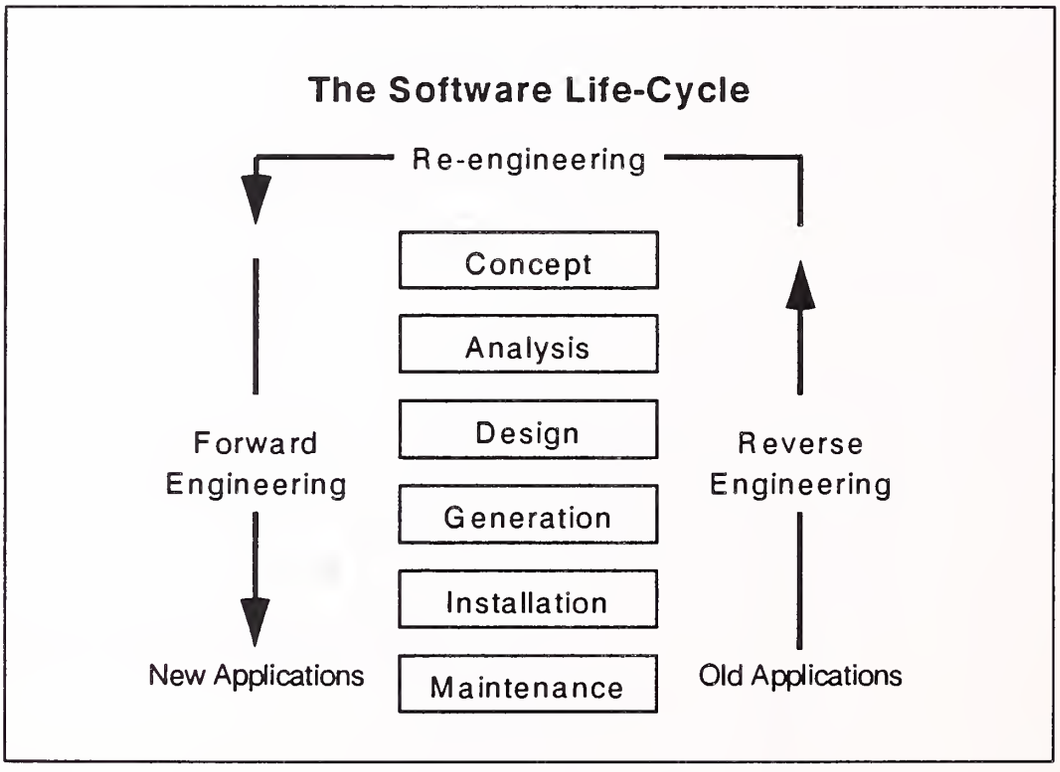
B

Software Engineering

Exhibit III-3 is one simple way of viewing the stages that a software project goes through from original concept to life-extending maintenance. Depending on the state of the software it can be considered to be following one of three essential processes:

- Forward engineering
- Reverse engineering
- Re-engineering.

EXHIBIT III-3



Forward engineering is the focus of most software and service vendors. It is the process of implementing a new or replacement application, starting more or less from scratch. Exhibit II-4 summarises it current status.

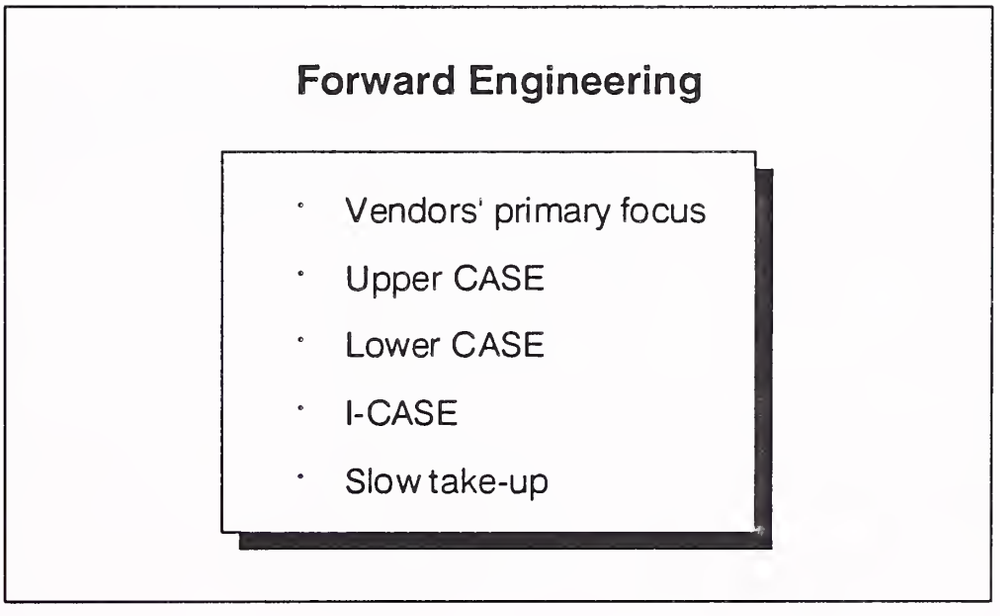
This process has had the benefit of a wealth of new development tools and methodologies over the last few years. To start with the CASE tools could be separated into two groups:

- Upper CASE - concerned with modelling and design of applications and the business requirements.

- Lower CASE - code generating from high level languages or specifications

More recently the term I-CASE has been coined for sets of integrated tools, commonly surrounding a shared repository. Despite the attractions of these tools at the workface, there has been significant reluctance in many IS departments to do more than experiment with them. It seems unlikely that such delays will make the decision, about what to invest in and how much, any easier in the future.

EXHIBIT III-4



C

The Development of CASE

For the last decade users have perceived the biggest weakness in CASE as "lack of integration". Dealing with integration - or, more usually, inadequacy in dealing with it - has been an important determinant of CASE progress and acceptance. The latest generation of CASE tools from the specialist vendors have adopted the class I-CASE or Integrated CASE.

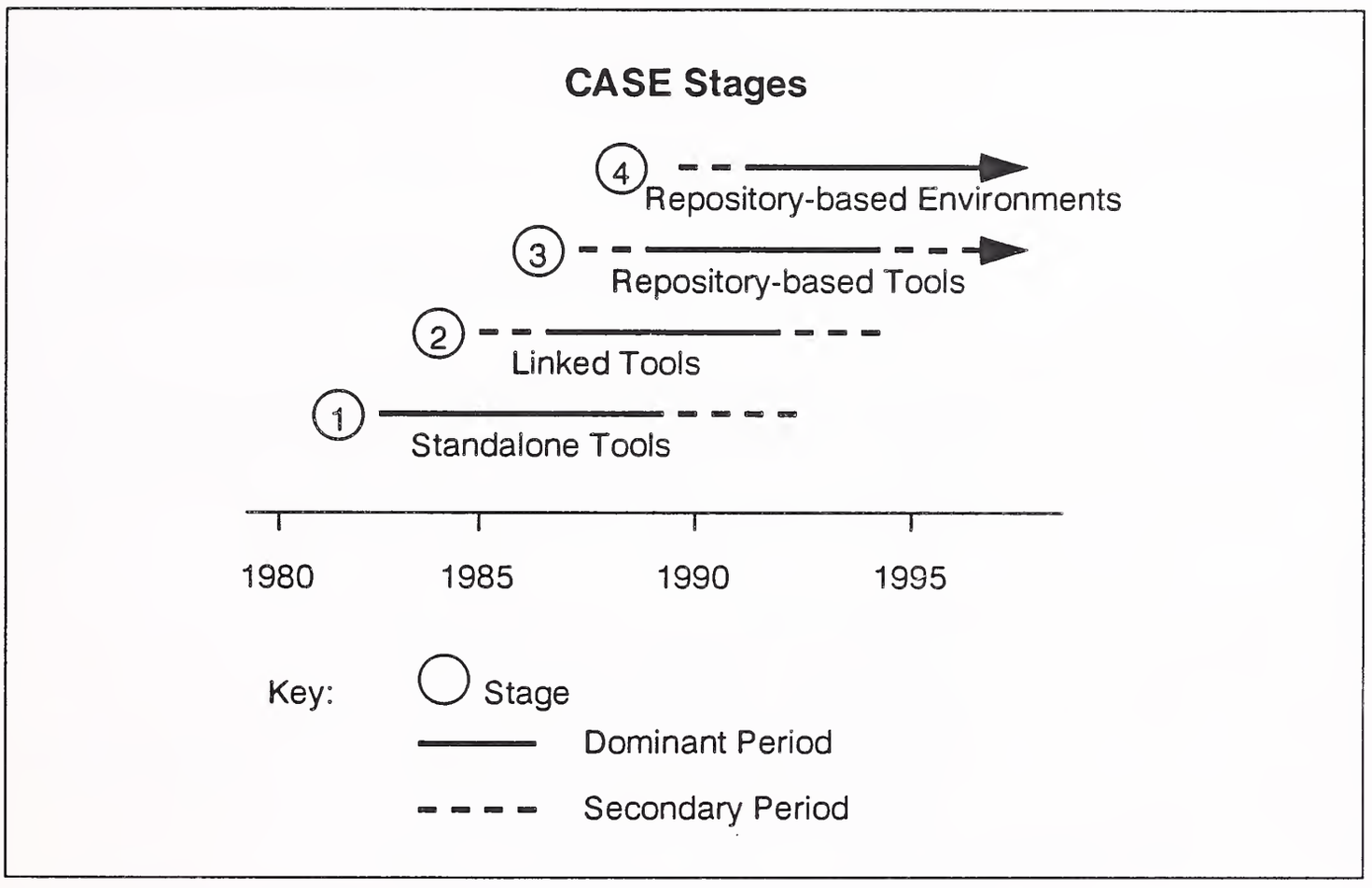
I-CASE attempts to bring together all the elements of resource management in an application development environment, not just the documentation of an application through all its stages:

- Project management
- Analysis
- Design
- Code generation

It achieves this through the use of a common repository, store or dictionary which holds all the documentation for the applications in an enterprise, and often the business models which they fulfil.

INPUT's analysis categorises CASE evolution into four stages, as shown in Exhibit III-5.

EXHIBIT III-5



Stage 1: Standalone CASE Tools

This is by now CASE's pre-history, but it is still important for understanding overall CASE trends. Both front-end and back-end CASE tools were important in Stage 1 (see Chapter 1 for definitions of CASE tools). However, front-end tools became especially prominent due to the following interrelated developments.

- Graphics-oriented workstations for representing data and process relationships became increasingly capable and inexpensive; originally, specialised graphics worksations were used, but soon standard PCs were acceptable.

- Information modelling methodologies became increasingly sophisticated as well as more practical.

- While it cannot be proved conclusively, it seems apparent in retrospect that the technological and conceptual developments were mutually reinforcing.

Stage 1 front-end technology could soon produce analysis and designs that were:

- Graphical
- Self-documenting
- Most importantly, sharable with non-technicians (e.g., business analysts).

Back-end technology during Stage 1 did not represent the potential breakthrough that Stage 1 front-end technology did. Back-end technology represented iterative improvements to traditional coding, but was still:

- Character based
- Procedural (in concept)
- Code oriented
- Inaccessible to non-technicians.
- Most importantly, the output of traditional code could be (and often was) independently maintained.

If the generated code could be maintained independently, then there could not be ironclad assurance that the application-as-documented (i.e. the generator input) would be the same as the application-as-modified (i.e., the working code). This represents the continuation of the age-old "patch" problems, where important changes are not coherently tracked. Library control is a fall-back position but is primarily an audit function rather than an assistance in development.

The largest problem with Stage 1 tools is that each was isolated from the other.

- It was up to the customer, with varying degrees of assistance from vendors, to tie tools together. Even if it was feasible for a customer to do so (and for all the largest IS organisations it would not be), this would rarely be a worthwhile use of resources.

- Consequently, during this period, CASE tools were almost always merely adjuncts to business as usual - perhaps producing pretty-looking documentation, but documentation just as likely to become instantly obsolete as traditional documentation.

- Stage 1 was the Golden Age of CASE shelfware, as customers found that making CASE useful was far more difficult than they had led themselves to believe.

Stage 2: Linked Tools

As Stage 1 developed, the defects inherent in having islands of CASE automation became clear to theorists, vendors, and customers (or potential customers). The most straightforward solution was to have the tool vendors take over the responsibility for developing links between tools for exchanging information needed for application development. In the mid/late 1980s there was a burst of announcements from tool vendors that they would support interfaces between one another. This was very desirable in that it enabled customers to focus on applications development rather than development of CASE linkages.

However, after a short time it became clear that announcing, or even initially developing an interface, was not a complete answer:

- There were no vendor-neutral information interchange standards.

- Information often had to be simplified (and value lost) in being translated from one dissimilar architecture to another.

These technical issues were serious and would make progress difficult at best. However, business-related factors were even larger stumbling blocks:

- The biggest problem was the sheer number of CASE tool vendors. At one point INPUT counted over 140 vendors offering at least twice that number of products. They faced virtually impossible challenges attempting to form linkages with each other:
  - How does one keep up with new versions
  - How should a partner be picked - on technical merit or market strength?
  - What if a potential partner is unwilling to cooperate?
- Some vendors formed semi-formal relationships, but most were promiscuous. Some marriages (i.e., mergers) occurred, but the total number of CASE tool vendors did not decline significantly.

Stage 2 brought no more order into the marketplace, and possibly less, than Stage 1. Customers (or more accurately, potential customers) were as confused and cautious as before:

- Some vendors were beginning to emerge as leaders, through some combination of name recognition, size, technical attractiveness, or market power.
- However, it was a rash (or very self-confident) IS department that would stake very much on a particular vendor (or combination of vendors) emerging victorious. To place a losing bet might well have meant wasted CASE development time and resources.

The risks in Stage 2 were often portrayed as opportunities; linkage, for example, was described as a chance for customers to select the "best of the breed" of different CASE tools. In a more mature market this might have been possible; as it was, Stage 2 was virtually doomed to failure from the start because of the enormous number of CASE products to choose from.

Stage 3: Repository-Based CASE Tools

Stage 3 also grew out of the frustrations with the isolated CASE tools of Stage 1. To oversimplify, linkage, rather than being secondary (as in Stage 2), was viewed as central to making the CASE concept function.

- Vendors stopped looking for a means of transferring information on data elements, data relationships, and logical processes between application development functions (i.e., CASE tools).

- Instead, information interchange became the centre of the CASE activities. This eliminated the complexities, redundancies, and synchronisation problems inherent in multiple linkages.

- The contrasts between the two approaches are shown in Exhibit IV-5. ("Repository" is used in Exhibit IV-5 because that term has the most currency). The repository concept has a simplicity and economy that would have ultimately made Stage 2 obsolete even if

  - There had been an order of magnitude fewer CASE vendors competing in Stage 2

  - IBM had not emphatically endorsed the repository concept (and made the investment to make it real).

The term repository is so closely identified with IBM and AD/Cycle that it is sometimes forgotten that IBM

- Was a relative late-comer in its public support of the repository approach

- Bought much of the core technology.

On the other hand, IBM provided a vital service by

- Producing a de facto standard for IBM platforms
- Accelerating the shrinkage in the number of CASE product vendors. The "noise" level caused by dozens of vendors in the marketplace has started to fall.
- Providing a stable target for customer planning.

Even in a repository environment, there will be some products that have tighter links than others.

- Certain products/functions will be supplied by the same vendor; these products will obviously work more in concert and be kept in better step developmentally.
- Some vendors, like those offering project management systems, may wish to support concurrent projects across different types of repository and hardware platforms. Their linkages to any one repository must necessarily be looser than would be the case for a product directed at a single repository.
- Methodologies are in a somewhat different position: current methodologies, by definition, pre-date repositories; consequently, a repository has to conform somewhat to existing methodologies. Methodologies have typically had no specific vendor sponsor. However, as time goes on particular repositories and closely associated tools may become implicitly more receptive to certain types of methodologies.

Current repository-based CASE is at least implicitly aimed at the forward engineering of applications. This places very real constraints on its applicability to solving real-world applications problems, which often involve a mixture of new development and modifications to existing applications.

Stage 4: Repository-Based CASE Environment

In some ways Stage 4 is a further extension of Stage 3, in the same way that Stage 2 was an extension of Stage 1:

- Stage 4 will be largely upwardly compatible with Stage 3.
- Stage 4 will represent a series of incremental changes.
- Stage 4 will not appear dramatically as a single announcement.

The "blending" of Stage 3 into Stage 4 will be represented by one or more vendors developing tightly coupled groups of tools *and* methodologies around a particular repository architecture.

- Where the repository design and execution is determined by a single vendor (e.g., IBM in AD/Cycle), at least one set of associated tools and methodologies will be offered - or at least tightly controlled - by that vendor.

- However, it will generally be in the interests of repository controllers to allow - and often encourage - third parties to provide alternative and niche offerings. This will provide limited-risk choices as well as a pseudo-open architecture for customers and other vendors.

The most visible addition in Stage 4 will be the linkage of forward engineering and re-engineering. Currently, re-engineering is isolated from the rest of CASE activities; as forward engineering and re-engineering are better coordinated, Stage 4 will take off.

However, with signs that growth in the software development market is becoming more and more difficult - competition is fiercer and spending is being cut - vendors are now looking to the re-engineering market.

D

Re-Engineering

As discussed in the *Definitions* section of Chapter I, there are several current words and concepts used to describe the re-engineering process (re-engineering, restructuring, renovating, and reverse engineering) in addition to older terms (e.g., corrective maintenance, adaptive maintenance, enhancements, etc.). There is not yet wide agreement within the industry on the precise meaning of these terms. There is not even the hint of loose consensus that exists regarding some of the forward engineering terms.

Until recently, the re-engineering process was straightforward: the objective was to fix an application and sometimes to re-write it; this was (and is) called maintenance. These objectives will not change, since some significant element of data processing must always be reactive to outside events (including program failure). However, much of maintenance will increasingly be viewed as re-engineering.

Re-engineering will involve two basic choices: *reverse engineering* or *re-use.*

Reverse-engineering will be somewhat analogous to maintenance as it is now, but with considerable change in emphasis:

- Multiple changes over time will increasingly take place using reverse engineered code as a starting point; much maintenance now is treated as if it were a one-time occurrence, even if similar one-time changes are made repeatedly.

- Reverse-engineered applications may have their life extended dramatically.

However, the full potential of re-engineering goes beyond the reverse-engineering and preservation of a particular application. Wider re-use of an application's constituents should prove to be equally valuable. This re-use can include the following:

- At the minimum, re-engineering technology can be used to understand the processes and data relationships in an application. This would be done preparatory to constructing a new application. For efficient communication the re-engineering and forward engineering should use the same conventions. Consistent conventions are needed because it may turn out that after inspection, the logic of the old application might be used to partially populate a repository.

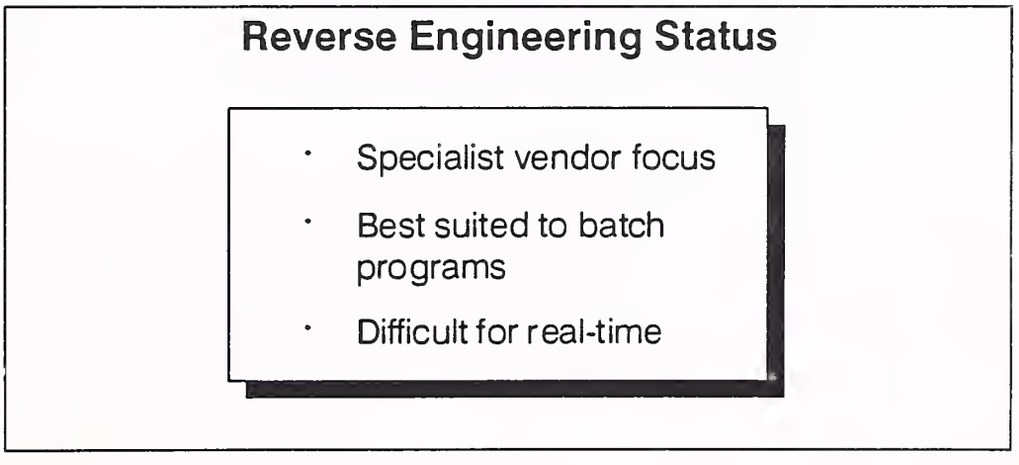
- Populating a repository from the logic in a previously written application can be a shortcut as well as a means of preserving the data processing "heritage" of an organisation.

- Finally, much larger pieces of an application can be used as the foundation for constructing an updated or expanded application.

- These steps form a continuum; the exact strategy to be followed is often not finally known until the organisation is fully engaged in the re-engineering process.

Exhibit III-6 summarises the status of reverse engineering at the present time. Bachman is probably the best known vendor in this sector.

EXHIBIT III-6



The extent to which existing applications are reverse engineered versus being re-used can have important implications for individual firms and for the CASE industry as a whole:

- If a very high proportion of existing applications are re-used, then CASE environments that are forward-engineering focused (i.e., Stage 3 CASE) will be less useful. (If a high proportion are reverse engineered, then forward-only tools are much more acceptable).

- Where a firm is highly committed to a changed technology base (e.g., client/server or enterprise information modeling), then re-engineered applications would only be cost-effective where short-term benefits predominated.

Both vendors and IS departments realise the importance of re-engineering; both indicate that there is a significant gap between the importance they place on re-engineering and their knowledge of it. Many vendors have already awakened to the implications of Stage 4 CASE, its requirements, and its opportunities.

Integrating re-engineering with the rest of CASE will occur in phases:

- The integration of forward-engineering components is well under way (Phase 1).

- Work is now in process by several vendors to take current standalone back-end re-engineering tools and link them to:

  - A self-contained front end (within re-engineering; Phase 2)

  - The back end of forward-engineering tools (Phase 3).

- Once there is a self-contained front end/back end within re-engineering (Phase 2), then it would be feasible to tie the front end of re-engineering to the front end of forward engineering (Phase 4). This would close the loop and begin to fully integrate forward engineering and re-engineering.

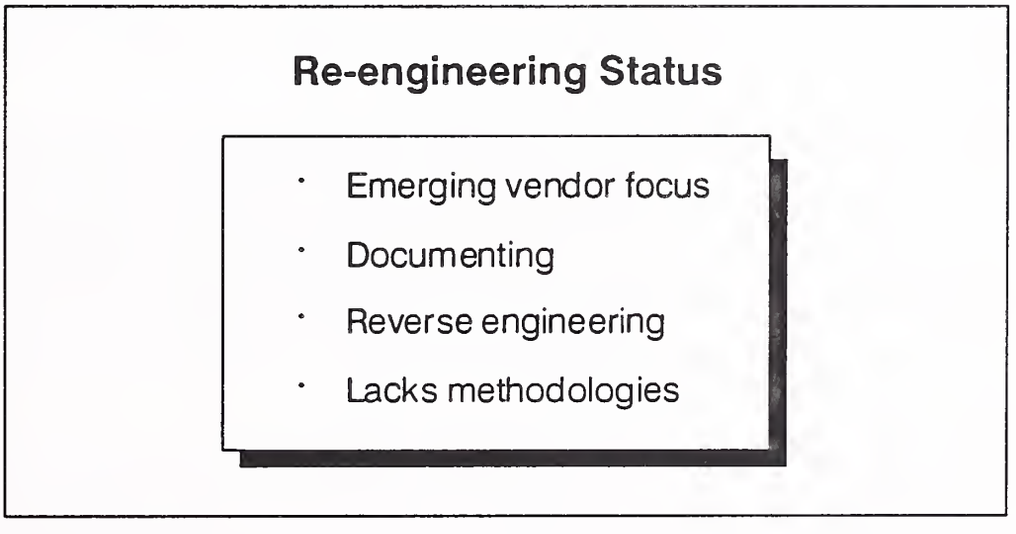
In 1992 INPUT expects little maintenance, modification, redevelopment and new development to be performed using re-engineering tools. This low usage is due to a lack of critical mass in re-engineering:

- Maturing tools that are still essentially standalone tools
- Re-engineering sponsorship by small vendors
- Lack of sponsorship by IBM
- Few methodologies; none widely accepted
- Little training available or used
- Low management priority given to maintenance.

By 1997 INPUT expects this picture to have turned around markedly - essentially because all (or most) of the factors above will have been reversed.

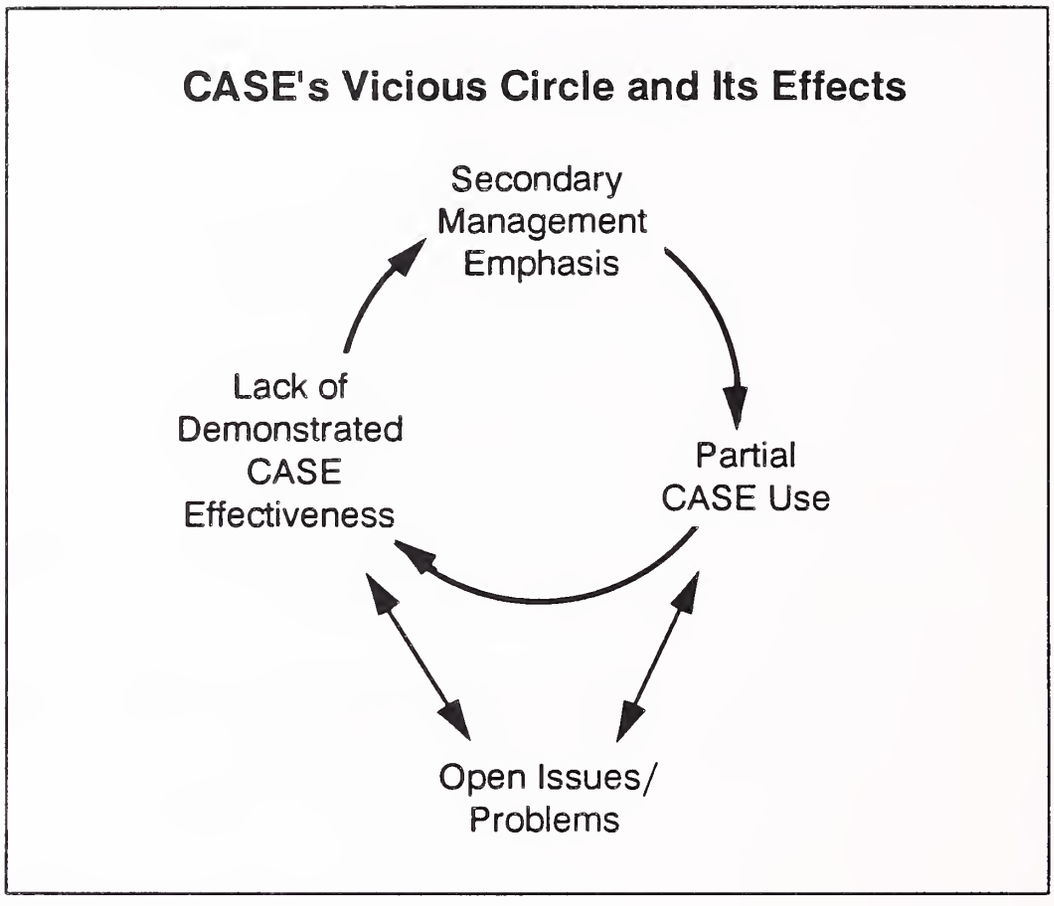
Exhibit III-7 summarises the current status of re-engineering in the CASE arena.

EXHIBIT III-7



CASE suffers from a vicious circle of inter-relationships between partial CASE use, lack of demonstrated CASE effectiveness, and the placement of often secondary emphasis by IS management on CASE issues. Exhibit III-8 shows these interdependent relationships which slow the adoption of CASE in many organisations.

EXHIBIT III-8



The arrival of re-engineering tools and methods, with which IS departments can start to tackle their operational software workload, should provide a strong incentive to IS managers to invest in highly practical tools rather than theoretical engineering principles.

IV
User Experience and Issues

This chapter addresses the results of an in-depth user survey consisting of face-to-face and telephone interviews, carried out in early 1992 in France, Germany and the United Kingdom. An analysis of the research sample of sixty organisations is given in Appendix A.

Interviewees were chosen, in relatively large organisations, from those with special management responsibility for systems engineering, application development, software quality, or IS management.

A
Purchasing Process

Decisions concerning the adoption of CASE tools are not easy. The investment extends well beyond the expense of the tools and workstations into lengthy training, introduction of new working practices, pilot projects, and so on. The response from users who had already bought into CASE identifies the key factors influencing the decsion process - Exhibit IV-1 lists the key considerations.

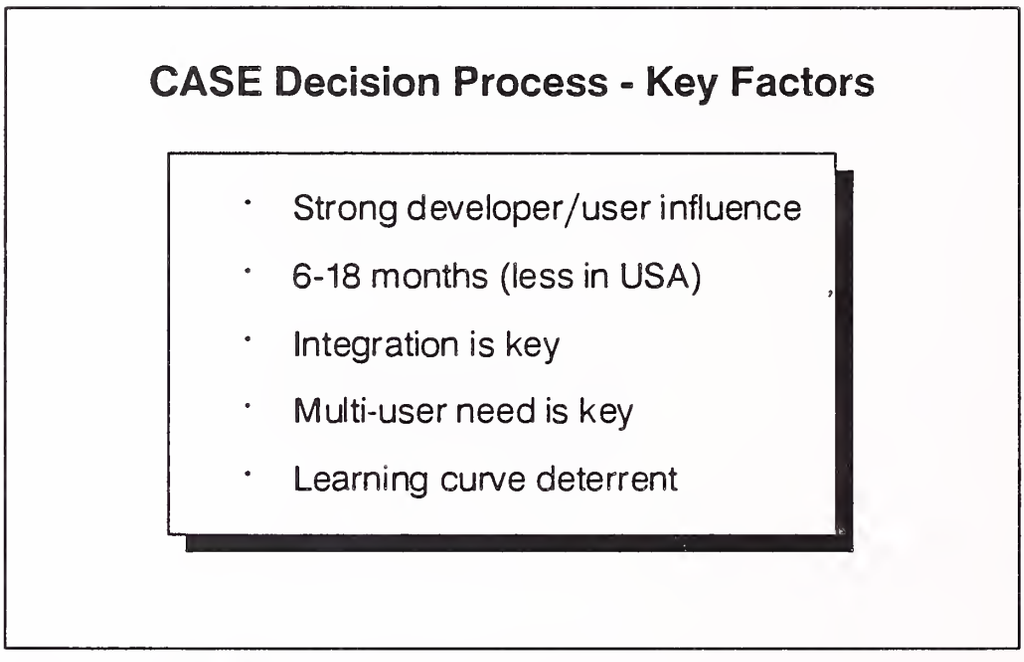
Several respondents felt that the actual users of the CASE tools need to be so committed to their use that their influence must not be under-estimated. Users must feel that it has been largely their decision. Several respondents talked of having a CASE tool forced on them and the resulting lack of commitment.

The high price of good tools and the need to make everyone party to the decision makes choosing CASE products a long process. Our research in the U.S. indicated timescales typically half those in Europe.

Few respondents were happy to leave CASE as a PC tool for individual developer use. The sharing of information and the ability to work as a team on projects are key requirements.

The initially high investment in product and in learning its use and new working practices, are key deterrents to faster take-up of these modern techniques.

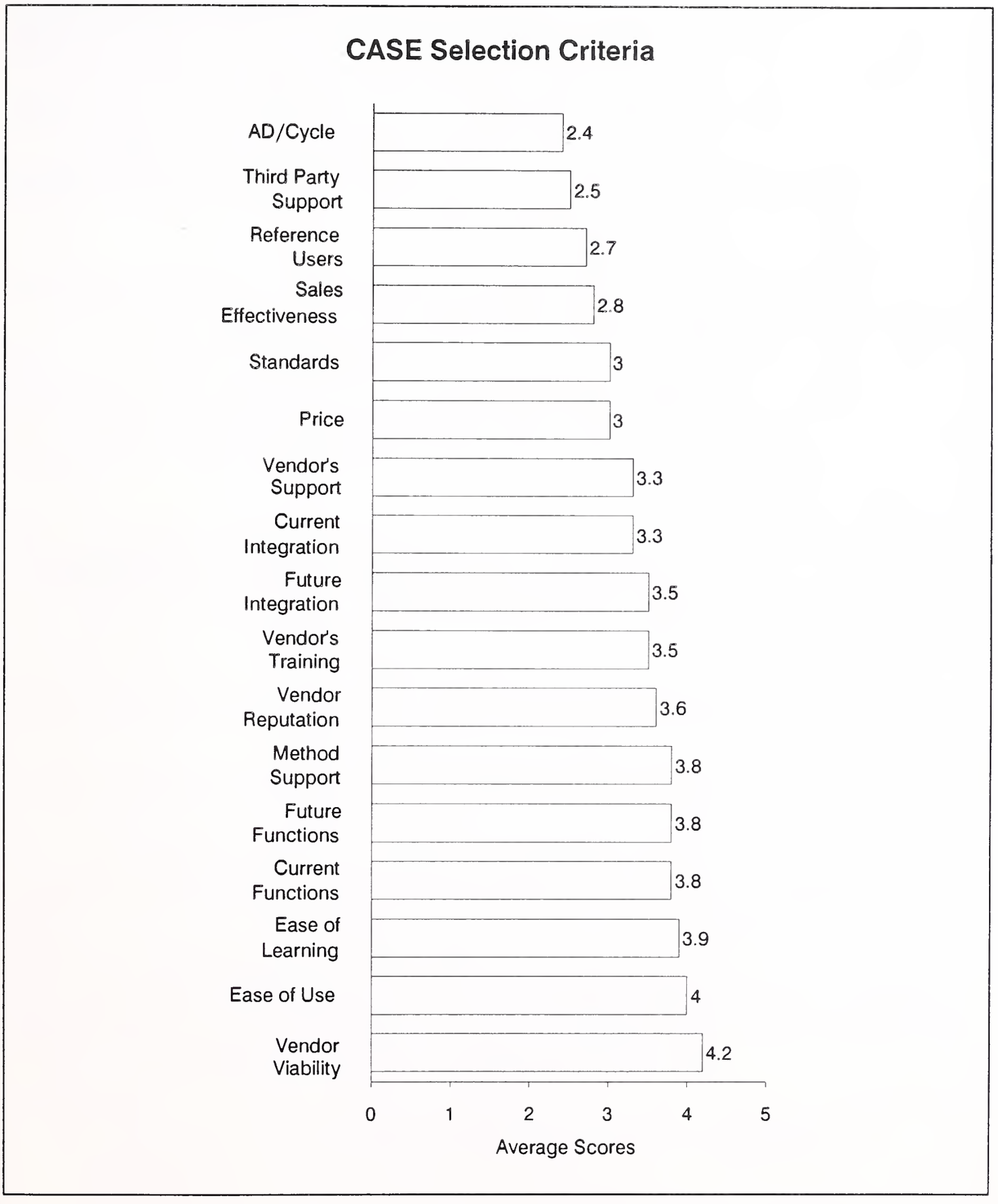
EXHIBIT IV-1



The choice of CASE vendor and product has been subject to some refinement by respondents since all those interviewed had already made such decisions. Few would change the way they went about it if they were going to do it all again. Those that would change would be more critical in testing vendors claims for product features.

Exhibit IV-2 shows the relative average scores in response to the question: How important do you consider each of these factors when deciding on a CASE product? (A score of 5 is very important, 1 is not important at all.)

EXHIBIT IV-2



The financial viability of the CASE vendor scored highest overall reflecting the extreme vendor dependence which users expect as tools and methodologies become essential parts of their working practices. The recent failure of some well known CASE vendors will have re-enforced this concern still further. Respondents were very clear that their adoption of products and methods for advanced application development environments would leave them potentially vulnerable to the fortunes of key vendors. This fact acts as a significant inhibitor to faster adoption of CASE.

Ease of Use and Ease of Learning both score very high in importance. The investment in training staff to use complex new tools and in understanding new systems engineering practices cannot be understated. This is one of the weakest links in any decision to adopt a CASE-based approach to systems engineering. Staff who have been used to the freedom of undertaking application development or support projects in their own way may find it difficult to learn whole new ways of doing their jobs.

A major surprise in the results is the low score for AD/Cycle conformance. The survey was conducted during early 1992, when doubt was being expressed strongly in the IT Press that AD/Cycle was ever likely to succeed. These results add weight to that view.

When responses in each country are compared against the pattern for Europe as a whole they show that among the most important factors:

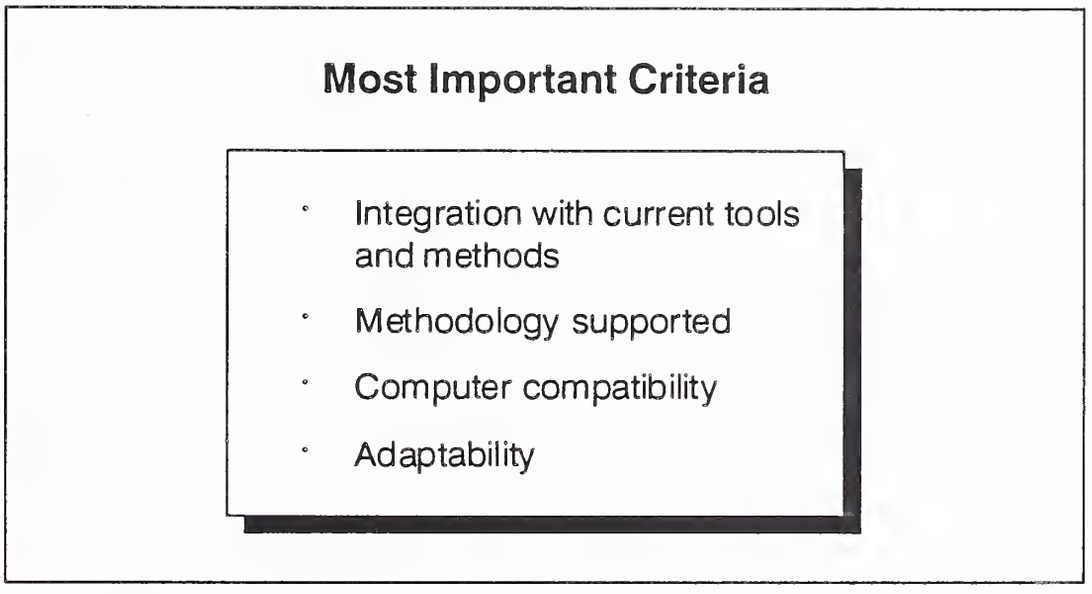
- German respondents felt that the vendors' training schemes were extremely important, much more so than French or British users.

- French were the most pragmatic about the importance of the products working functionality, putting this as their top priority factor.

Before being prompted with a large list of factors, interviewees were asked to identify the single most important factor in their choice of CASE vendor. The result very clearly put integration as top priority, followed by compatibility with their chosen methodology.

Exhibit IV-3 shows the priorities.

EXHIBIT IV-3



B
Impacts of New Architecture

There were three areas of future change which respondents were able to identify:

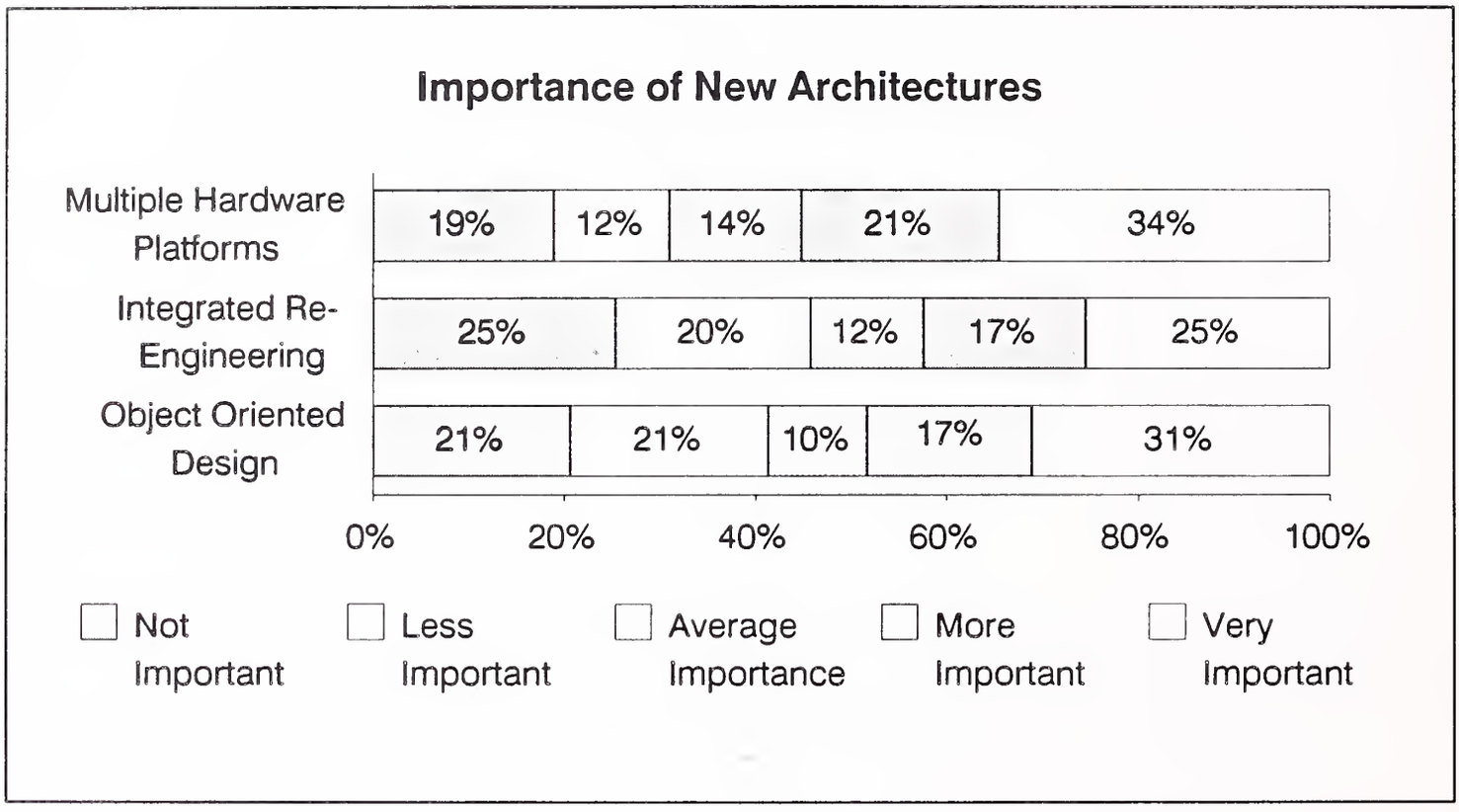
- The use of multiple hardware platforms as the target machines on which to run future applications.
- The integration of re-engineering into the CASE-supported development cycle.
- The use of object oriented software designs to optimise re-use of Exhibit IV-4 shows what percentage of respondents considered each factor to be important either now or in the near future.

Over two thirds felt multiple platforms were important, reflecting a wide recognition that distributed applications, with or without client-server architectures, are here to stay.

Integrated Re-engineering was less well accepted, or even understood, but still over half thought it important to very important.

Object orientated design was seen by nearly 60% as important, with half of these seeing it as very important.

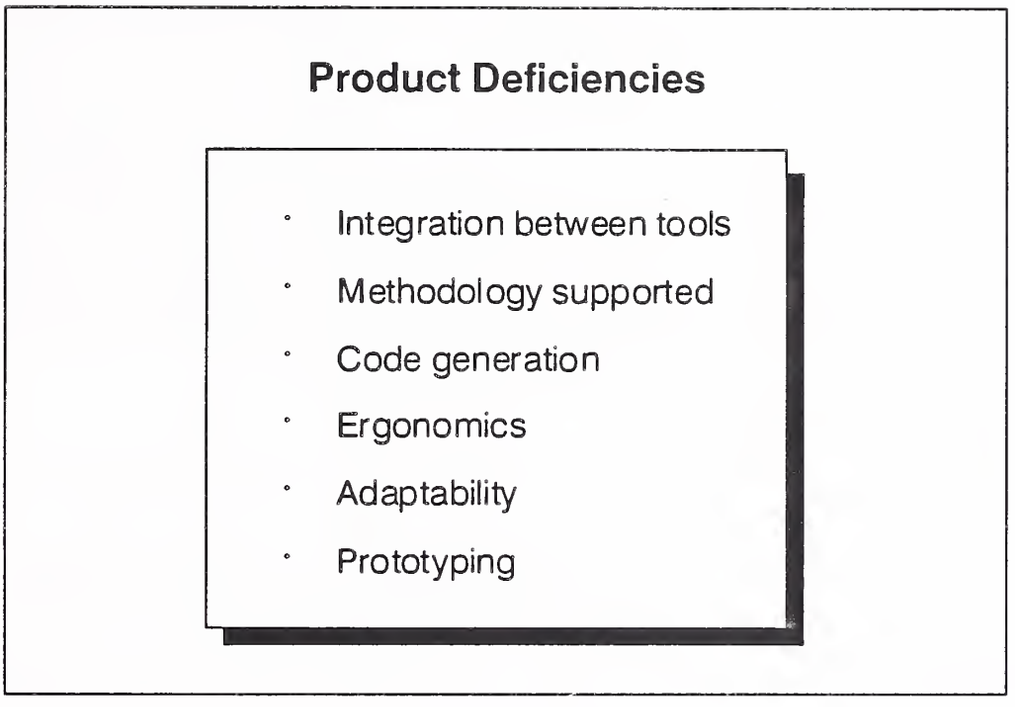
EXHIBIT IV-4



C
Product Deficiences

The low visibility of re-engineering as a CASE-supported activity resulted in not one mention of it as a deficiency in current products. Those aspects which clearly got several mentions in response to the question "what do you see as the biggest deficiencies in the current generation of CASE products?" are shown in priority order in Exhibit IV-5.

EXHIBIT IV-5



D
Benefits

Why should anyone invest in CASE products? This is obviously a key question driving the development of the whole market. Significant levels of investment had been made in the last few years by interviewees, the average being $730,000 spent to date.

An extraordinary variety of answers were given to the question "What do you expect to be the principal short and long term payoffs from CASE?". There is no single over-riding result driving these complex decisions. For the purposes of analysis the answers were categorised into six primary concerns:

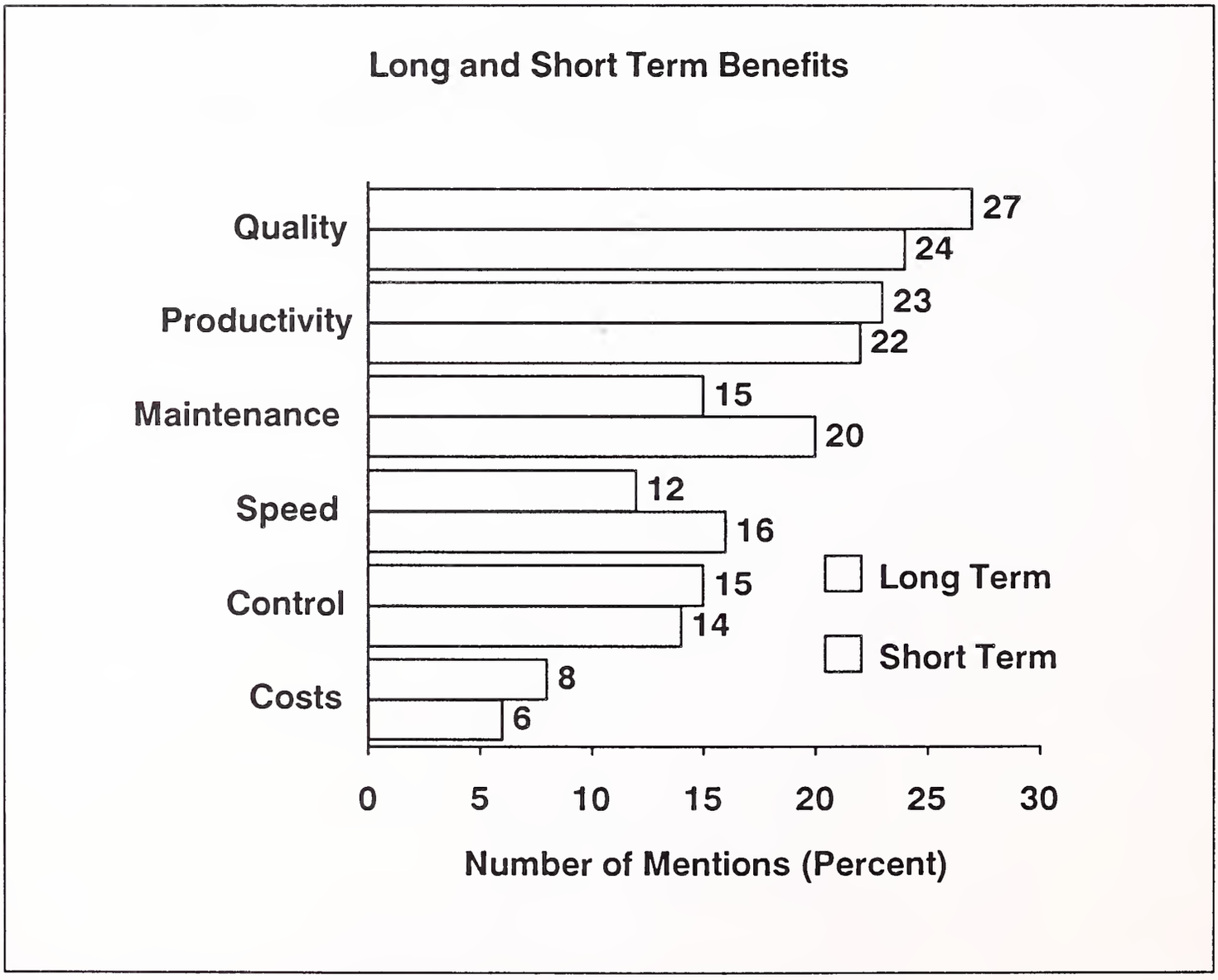
- Quality improvements of all types
- Productivity gains for developers
- Improvements to the maintenance process
- Speedier response to user needs
- Better management control
- Reduction in costs

In Exhibit IV-6 quality, productivity and maintenance are the areas in which respondents expected the most benefit. Cost reduction was expected in the short term by only a handful of respondents.

In the long term the expectations change. Quality improvements take a clear lead, but maintenance expectations fall.

There were clear indications that many respondents were applying CASE to their documentation problems. Using the tools to establish consistent documentation of existing software. This, in turn, was expected to give early improvement to their software maintenance workload. In the longer term maintenance was then much less of an issue than productivity and quality across the whole application development environment.

EXHIBIT IV-6



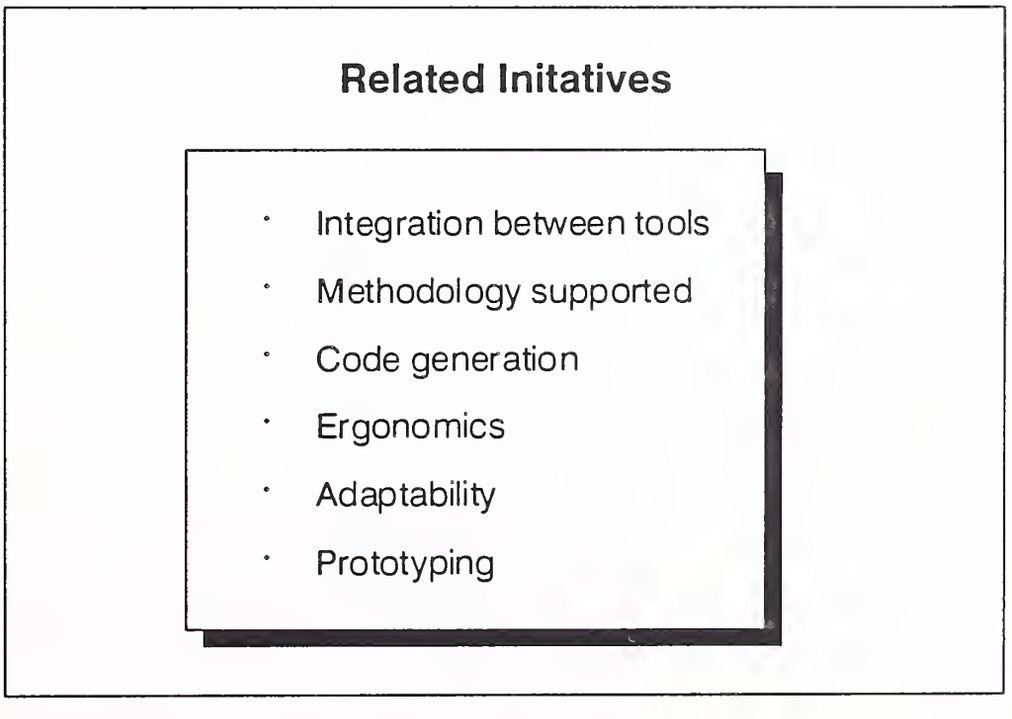
These six areas of benefit are all closely inter-related. "Quality" could be considered as a term which encompasses all five others. For vendors addressing this market it will be important to establish:

- Which benefit/requirement is key for each prospective client - e.g., is the client's key concern to measure and improve productivity?
- How to package CASE products as a solution to particular needs - e.g., present a break-even model for reducing maintenance costs.

E
Related Issues

Decisions on CASE can rarely be treated as independent of other issues. CASE products and skills are merely one element of a more complex application development environment. The most important issues, as perceived by respondents, are listed in Exhibit IV-7.

EXHIBIT IV-7



Most closely related to CASE are the methodologies used for system development and maintenance. These vary significantly from country to country and from organisation to organisation. Some respondents have developed their own methods and working practices, others have adopted the national standards such as Merise in France or SSADM in the United Kingdom.

Many interviewees pointed out how difficult it is to introduce a methodology *after* having made a major CASE purchase. Methodologies usually offer some freedom in choice of CASE product, but frequently the reverse is not true. Many CASE products cannot be integrated into a variety of methods.

Adoption of standard software platforms or techniques can also restrict the choice of CASE products. After considering over 300 potential products, one respondent found only a handful of products which would support their strategy for co-operative processing.

Adoption of new tools and methods cannot usually be done effectively without some re-organisation in working practices and reporting structures. Just as a business must re-consider its business processes in order to exploit IT fully in the market, so an IS department must re-consider its application development processes in order to exploit new technologies and serve its business clients more efficiently.

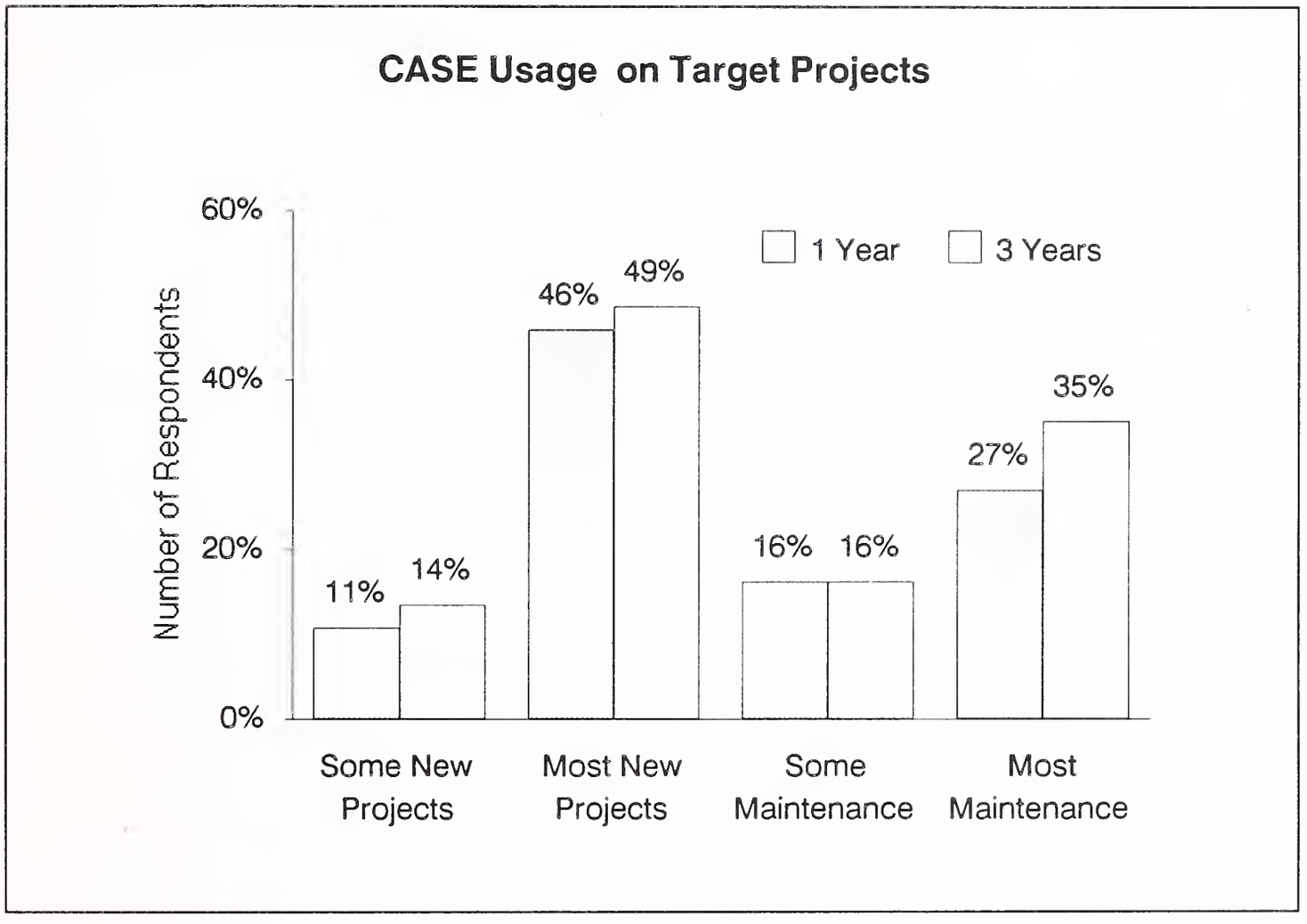
The advent of CASE-based application development has raised a new issue in the area of training. Software development staff are not only being asked to change their working practices, they are being asked to extend their skills and become fully qualified systems or software engineers. This is a daunting task for them and their management. A large element of their current skill has been learned on the job within an IS department, often in a rather ad hoc way. Turning professional staff into information engineers requires a major commitment to training.

F
Planned Usage

The sheer complexity of moving towards a new, highly formalised application development environment using information and systems engineering working practices, slows the spread of CASE within most IS departments.

There are a number of choices management can make in terms of how widespread they intend the use of CASE to become. Should they limit it to pilot projects? Should it be confined only to new developments? Exhibit IV-8 shows the way they initially use their CASE products compared to the types of project they expect to use them on in three years time.

EXHIBIT IV-8



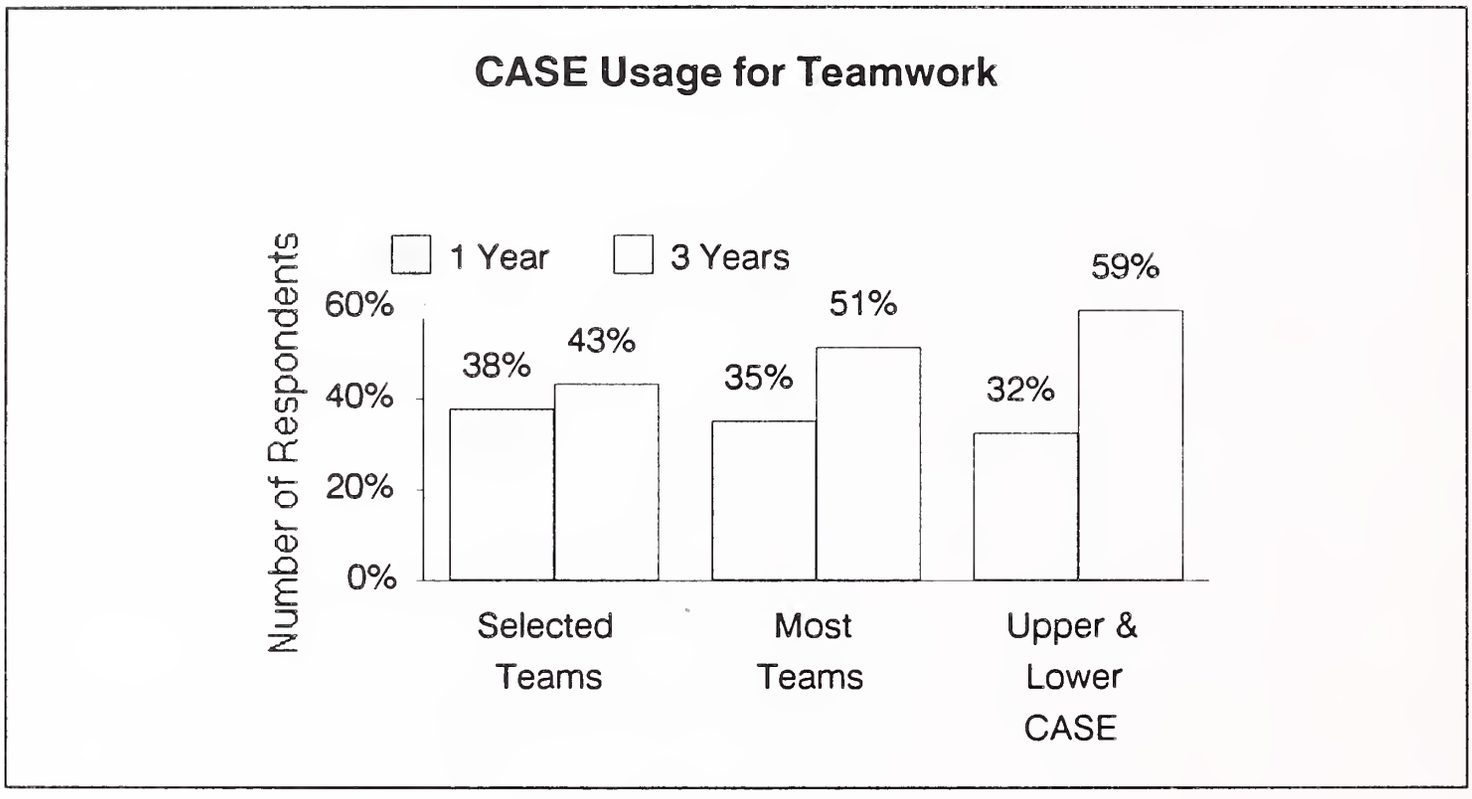
The most noticeable trend is the increase in use of CASE to improve maintenance projects. Although maintenance was shown earlier to improve through the use of CASE to enhance software documentation, it still tends to be one of the last areas in which widespread use will become the norm.

In other words improving documentation can already be achieved - to some degree by the adoption of reverse engineering techniques. But then using CASE to re-engineer such software, or merely to introduce modifications is still not common practice.

Two other trends in the usage of CASE are illustrated in Exhibit IV-9. This shows how CASE is moving from tools for the individual - largely based on PCs or Workstation - towards tools for teams. Over half the respondents see the use of CASE spreading to most development teams in their organisation within three years.

Also shown is a huge jump in the use of both upper and lower CASE tools. More than any other factor this shows how CASE is becoming essential to the normal operations of any large application development department. It is no longer a question of whether to adopt a CASE strategy, only a question of when.

EXHIBIT IV-9



G
Re-engineering Expectations

Re-engineering is the term applied to carrying forward an existing investment in application software to new hardware, software and development environments. The promise of re-engineering is that it will reduce the costs of software maintenance and improve responsiveness to changing user needs. When will it start to fulfil this promise?

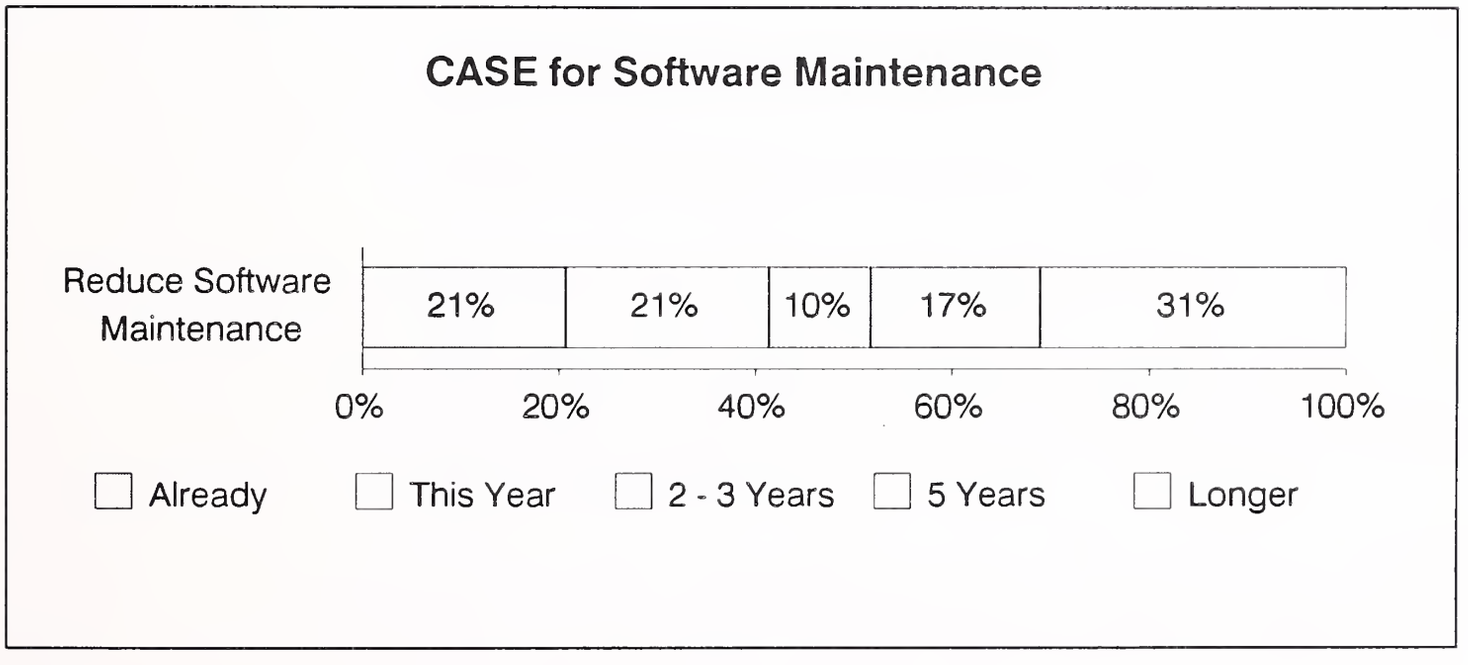
To some extent it already has started. Exhibit IV-10 shows the spread of responses to the question "When do you expect your use of CASE to begin to reduce your software maintenance costs?" Over 40% say it will have started making an impact during 1992.

However the Exhibit also shows that a significant one third of respondents are very sceptical that they will ever reduce maintenance costs.

This parallels the results of research INPUT did last year into software maintenance, where the main obstacle to improvement was the lack of awareness among IS management that there were improvements to be made. For example there was little awareness of the possibility of outsourcing maintenance, the use of reverse engineering or re-engineering CASE tools, or the application of maintenance methodologies and working practices.

Last year's study suggested that the crux of the software maintenance problems was a lack of measurement and management processes within IS departments.

EXHIBIT IV-10



V
CASE Market Forecasts

INPUT forecasts that the CASE market in Europe is likely to grow from $345 Million in 1992 to $1.2 Billion in 1997. Over the five year period this represents an average compound annual growth rate (CAGR) of 28%.

CASE's market future will be heavily influenced by:

- Organisational readiness among IS departments and the effects of recession in the near-term.

- Development in re-engineering techniques in the medium-term.

Two alternative scenarios are presented in Exhibit V-1.

- The "high" alternative assumes that IS departments become organisationally more responsive and the CASE vendors offer more integrated product lines. This alternative is expected to have a 25% chance of occurring.

- The "low" scenario encompasses a lack of further advances in CASE. CASE will not assume a strategic role and will be oriented primarily to technical staff only. This alternative also has a 25% chance of happening.

- The most "likely"scenario - with 50% chance of succeeding - assumes adequate progress in acceptance of CASE and development of re-engineering tools and methods.

EXHIBIT V-1

CASE Product Forecast Scenario Europe

Europe	1991	1992	1992-1997 CAGR (Percent)	1997
High	300	390	35	1780
Likely	300	345	28	1200
Low	300	320	15	630

Country variations are shown in Exhibit V-2. Germany can expect the fastest growth, starting from a smaller base than France or the U.K., with a CAGR of 37% between 1992 and 1997. The major inhibitor in Germany is the lack of any nationally adopted standard methodology.

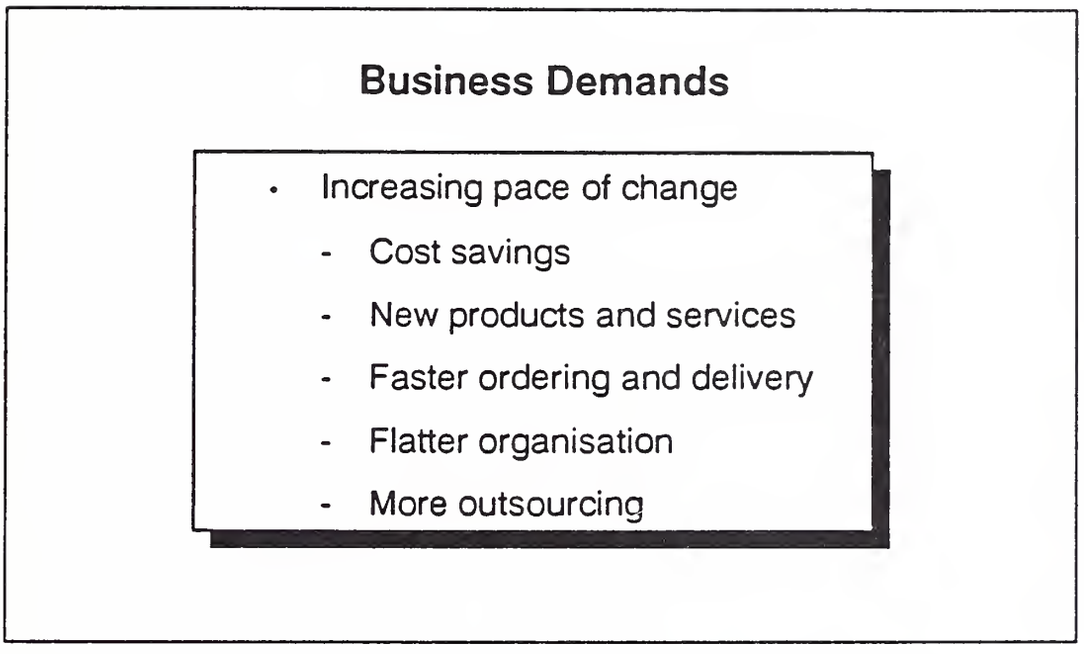
EXHIBIT V-2

CASE Product Forecast by Major Country

Country	1991	1992	1992-1997 CAGR (Percent)	1997
France	90	105	27	350
Germany	65	80	37	385
United Kingdom	105	115	23	320
Rest of Europe	40	45	26	145
TOTAL	300	345	28	1200

The factors which will drive growth are the ever increasing demands of businesses on their information systems - highlighted in Exhibit V-3 - resulting from the pace of change in practically every business sector.

EXHIBIT V-3

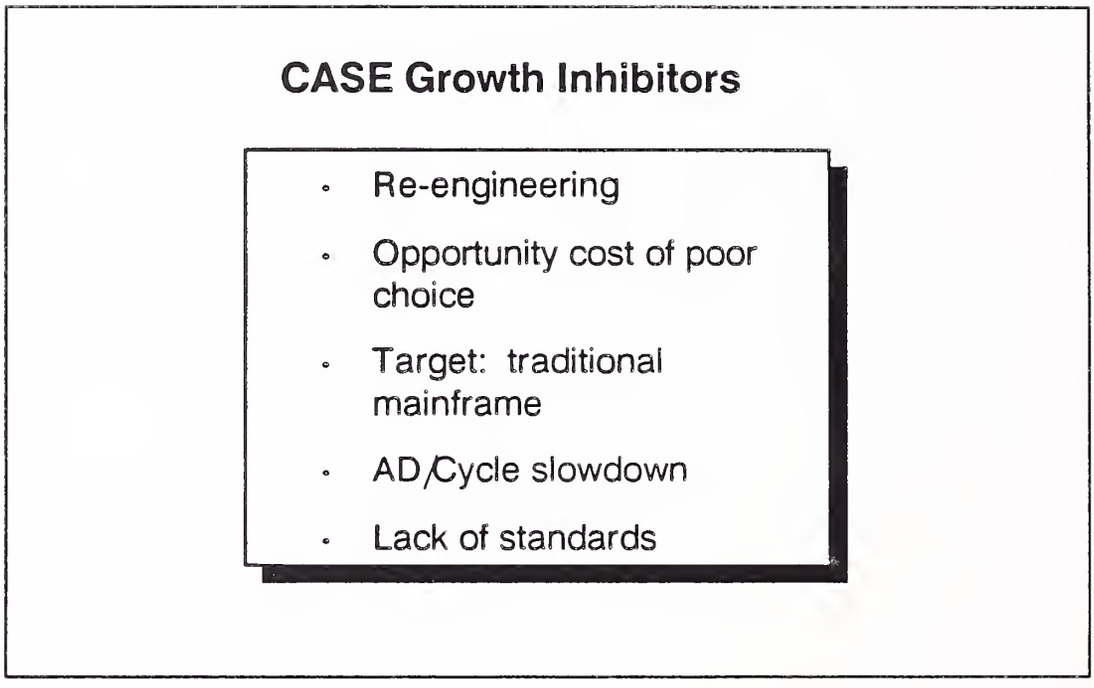


Powerful forces are at work in the CASE market. Many of them slow the development of the market. The important inhibiting factors are listed in Exhibit V-4.

The latent demand for re-engineering techniques may be huge, but major vendors have only started to address the issues seriously in the last couple of years. While the market for tools and methods aimed at new applications was growing rapidly, re-engineering was largely ignored. Now that IS spending growth has hesitated and competition has become cut-throat, re-engineering is winning attention from vendors of both tools and services.

Equipment manufacturers have recently embraced re-engineering as it also promises a smoother path from proprietary platforms to open system standards. So re-engineering is a "new" way for equipment vendors to retain customer loyalty while encouraging migration to new platforms and discouraging migration to new vendors.

EXHIBIT V-4



A strong deterrent to investing in CASE is the volatility of the market combined with the high entry costs (products, staff training, new working practices, organisation changes, etc.). A wrong choice could prove expensive - perhaps more expensive than delaying any decision.

The target for CASE is control of the life cycle for large complex applications - traditional mainframe software. But downsizing, PC networking and the open systems trends are all casting doubt on the future needs in a mainframe environment. Will applications which are implemented on distributed small systems require the same management controls and methodologies? Will the problems just fade away? The possibility of fragmented small systems and distributed responsibility for them is another reason - probably false - for delaying investment in CASE.

AD/Cycle seems to have got off to a false start. IBM originally conceived it as a mainframe environment. The attractions of distributed, downsized architectures has forced IBM to re-think AD/Cycle and acknowledge that it must cater for networked systems and repositories. Delays in bringing product to market has slowed CASE take-up.

New CASE products are announced every week. They are an essential enhancement to nearly every other major software product as well as products in their own right. But the lack of standards in two areas is a crucial deterrent:

- There are no independent standard interfaces which would allow CASE tools to be readily integrated together.

- There are no European or international system/information engineering methodologies, only one or two national favourites and a plethora of in-house and proprietary methods. (The EC initiative, Euromethod, may soon provide a framework for existing methods.)

Factors working in favour of CASE market growth are shown in Exhibit V-5. As the first point makes clear, many IS managers do not see that they have any choice but to embrace CASE, even though they know that they will become absolutely dependent on it as time goes on - CASE is considered essential at the large enterprise level.

Until something replaces it, AD/Cycle is expected to provide the common factor between different CASE products in the IBM environment. Other repository-based tool sets tackle the need to cater for mixed-vendor hardware environments. These include CASE products such as:

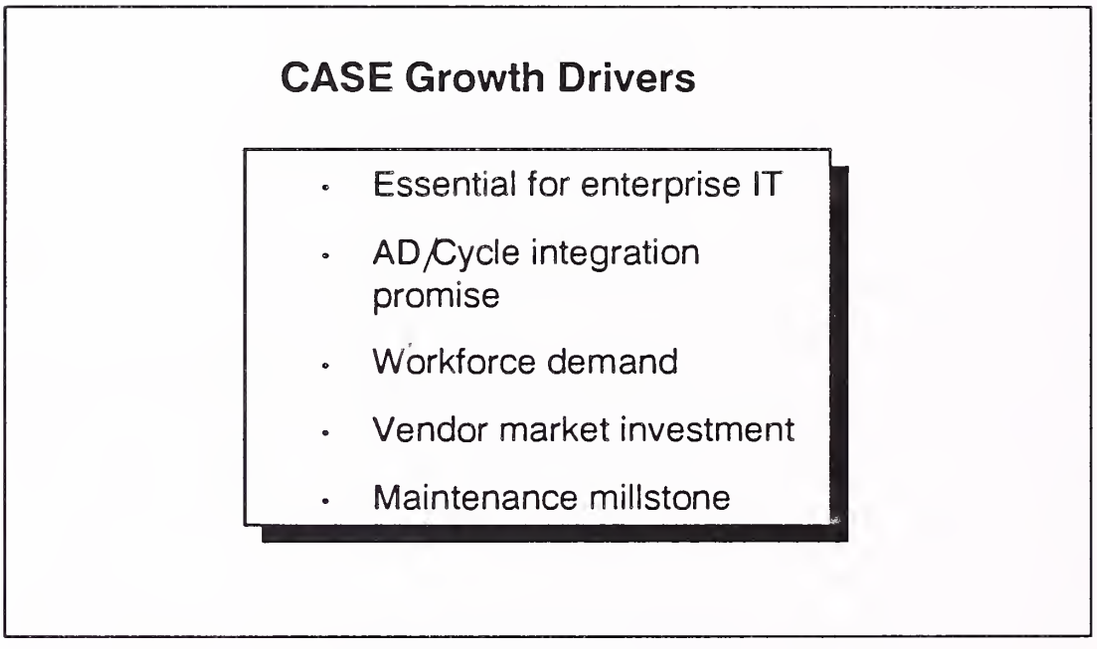
- Andersen Consultings' Foundation
- CGI's PACBASE
- Knowledgeware's ADW
- LBMS' Information Manager
- Oracle's CASE*
- Sema's Merise II
- Softlab's Maestro II
- Software AG's Predict
- Texas Instruments' IEF

Demand for new technology software tools from the users - consultants, designers, analysts, programmers, project managers, etc. - is extremely strong. Investment in good tools for such staff can be highly motivating. Lack of such tools can similarly demotivate.

The role of vendors in creating market demand and expectation should not be underestimated. Unless vendors continue to invest in priming the market, and removing confusion, the inhibiting factors may slow demand to a damaging extent.

Maintenance still takes second place to new applications in most peoples' priorities. But it is the awareness of the mess which most IS departments have created over the years - through poor working practices, lack of engineering discipline and obsolete software - which drives IS management to invest in improving the situation in future.

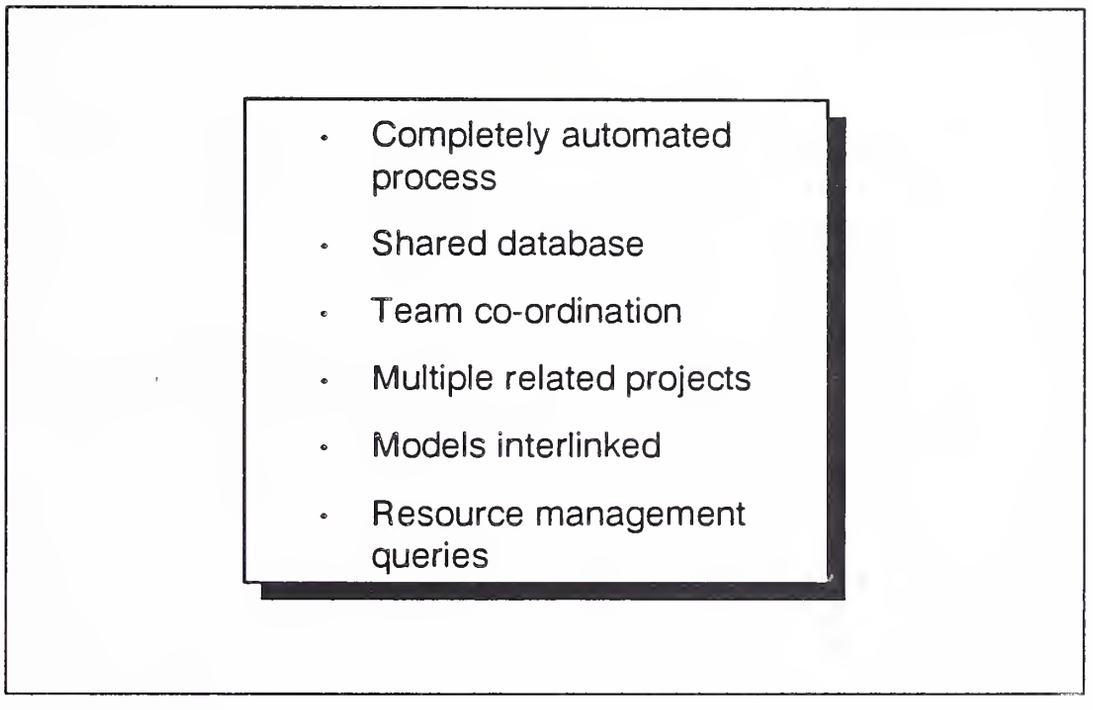
EXHIBIT V-5



Integration is probably the most widely used term in the IS industry. In the application development environment it also has a confusing range of meanings. Exhibit V-6 identifies some of the different meanings exposed when interviewees were questioned further as to what they meant.

EXHIBIT V-6

Integration - Mixed Targets



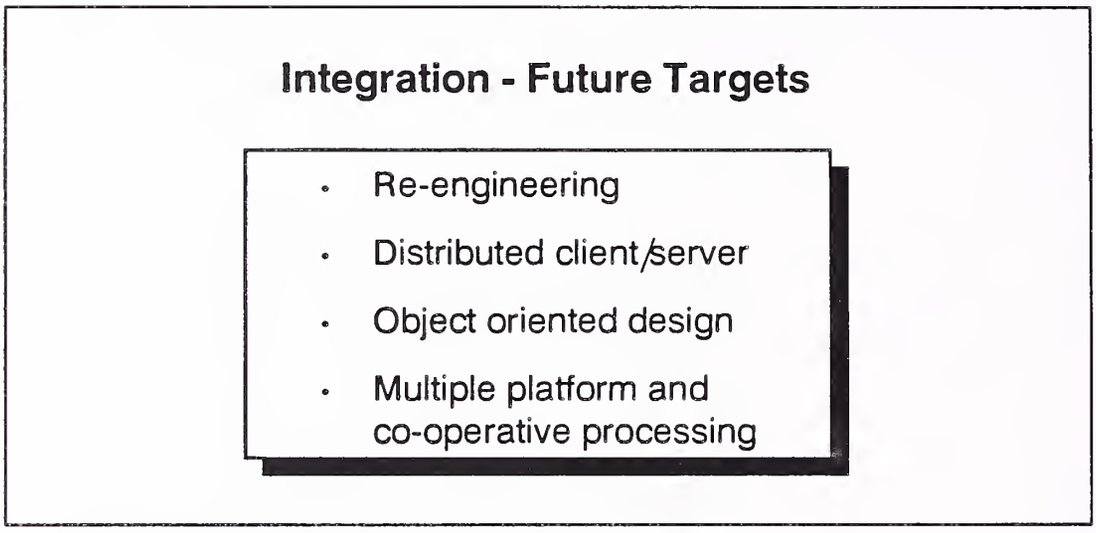
In essence integration refers to the ability to get greater value from CASE tools, models, projects and resource management. The requirement results from lifting the objectives for a tool away from the individual workstation up into the operation of a team and then up to managing the systems engineering for a whole enterprise.

From senior IS management viewpoint the requirement is integration in terms of sharing information, tools, skills and working practices. CASE tools form only one element of this, but I-CASE promises to bring it all together into a uniform whole.

The future objectives for I-CASE, for purchasers and vendors alike, are shown in Exhibit V-7.

Re-engineering techniques, methodologies and tools are required to protect essential investments in existing applications and carry them forward at minimum cost.

EXHIBIT V-7



VI
Conclusions and Recommendations

There is no doubt at all that the market for CASE products will continue to grow. I-CASE will cater for those who need tighter control over the development and maintenance processes at enterprise or project level. More pragmatic CASE tools will become standard features of most other software product, providing templates, models and procedures which simplify the creation and maintenance of applications, linking them directly to their business requirement.

Growth in re-engineering is much more difficult to predict. Will users choose to migrate their old applications to new software platforms? It is heavily dependent on IS management being able to value the underlying, hugely complex business processes embodied in such old applications.

At the simplest level it is a matter of answering the questions:

"Is the current wealth of application software, or even its underlying design, worth preserving?"

"Are we needing to re-engineer our *business* so much that existing applications will need to be changed out of all recognition?"

A

Growth in Software Reengineering

The primary factors inhibiting the growth of software re-engineering are:

- Downsizing
- New Architectures (e.g. OOPS)
- Business Re-engineering

The trend to downsizing and open systems is encouraging the development or purchase of re-usable, scalable applications. Only a few of these are going to be based on existing designs.

Tools and techniques for developing applications are still improving rapidly. Most users have their eyes on Object Oriented software, even though it may be several years before it will be used widely. It is unlikely that applications designed without Object Oriented software (OOPS) in mind can be re-engineered to that architecture.

There is a growing realisation that many IT projects have resulted in businesses "automating their problems". Re-addressing the way they do business and re-engineering the business to make best use of IT are attractive radical options. Few old applications are likely to survive for long and fewer perhaps will get re-engineered to fit the new regime.

The factors driving growth in software re-engineering are:

- Investments not yet written-off
- Software tools:
  - Migration Aids
  - Reverse Engineering Tools
  - Portable Software Platforms

The feasibility and growth of re-engineering depends heavily on:

- A clear recognition that the value of existing applications lies in their proven structure, the implied user working practices and users' familiarity.

- The availability and adoption of life-cycle CASE tools for reverse engineering existing poorly documented applications, updating their functionality, and then forward engineering them for re-use on modern software platforms.

- The development of methodologies which properly account for a pragmatic re-engineering process, rather than assuming an enterprise business model already exists or that applications are best given a fresh start from a clean sheet of paper.

Re-engineering is an attractive process for software or system vendors with large installed bases. The advent of open systems and downsizing continually threatens to lose them customers to their competitors. Re-engineering in the form of cost effective migration tools and procedures can minimise this threat, keep customer loyal, and allow them to move to new software platforms when appropriate. These vendors have a vested interest in helping their clients carry forward their pas software investments.

In the past there has been much talk of the applications backlog - considered by some as a statistical myth. The real backlog might well be all the business-critical applications code that exists in poorly-documented form, written over the past three decades. A burden which consumes valuable resources just to maintain. CASE tools and methods which can help reverse engineer old code and re-generate documentation and designs are an almost essential prerequisite for re-engineering growth. The number of such tools is multiplying rapidly.

More recently developed applications may benefit from being based on portable and scalable software platforms like Oracle's database. Problems of migration to new hardware environments are largely handled within the software platform rather than at the application level.

B
User Recommendations

IS managers need to carefully assess the choices open to them in their applications investment. The choices to be made for each existing application can be summarised as:

- Drop - benefit no longer justifies cost
- Hold - minimise support and maintenance
- Carry forward - improve service to users
- Re-new - replace with new system by:
  - Re-engineering and migrating
  - Developing new custom application
  - Buying suitable application package

Each of these routes must be assessed in terms of:

- the level of service provided for and needed by users
- the contribution the application makes to today's business
- the costs of alternative choices concerning its future

The ideal of an application development environment where changes can be implemented at the tough of a button seems as far off as ever. But without the use of integrated CASE tools and clear engineering methodologies, the increasing complexity of IS systems is going to make them even more difficult to manage.

C
Vendor Recommendations

Re-engineering offers software and services vendors a set of opportunities which they have been able to largely ignore in times of rapid IS market growth.

There are several types of vendor active in this market:

- The I-CASE vendor expecting to offer a complete tool-box approach to the process

- The product vendor with powerful forward engineering tools, whether Upper-CASE or Lower-CASE.

- The reverse engineering CASE vendor targetting environments such as COBOL.

- The methodology vendors, most of whom have little focus on re-engineering.

- The professional services vendor who has his own standards but often has to adopt his clients' chosen methods.

For the CASE vendor there are two main challenges emerging as re-engineering becomes a financially viable oprion for users:

- How to package up the tools which can reconstruct the specification and documentation of existing business critical applications for their user base of clients. This form of reverse engineering option can offer an early payback to the client by merely improving his ability to manage the maintenance workload.

- How to then re-engineer and migrate the most valued of these old applications into the environment used for new application development. Payback for the client is likely to much longer term than the reverse engineering phase. Those vendors who primarily sell a software platform can be expected to have an advantage over the purely CASE product vendor.

The I-CASE vendors will need to extend their field of influence in order to support users with an ever more comlex mix of tools and applications. There is a strong need for resource management tools, to allow IS management a fuller view of the projects and staff under their management. The complexity of this task is not expected to ease as a result of downsizing or distributed systems.

In general, CASE vendors will probably need a wider variety of business partners in order to address the re-engineering market opportunity. It will be essential to be able to understand the financial implications of all the different choice each client faces in determining the most effective way to engineer an application - see section B above.

Specialist or niche vendors need a set of established partners to meet users need for all complex aspects of the re-engineering process. These include having the skills to:

- Value existing applications within a business
- Cost alternative approaches to up-dating applications (re-engineer, re-invent, buy-in, etc.).
- Supply reverse engineering tools
- Integrate the new documentation with other enterprise models using latest methodologies
- Support the new life-cycle process.

These skills range from those traditionally considered management consultancy through to detailed platform software. Most product vendors will partner a range of professional service organisations.

For the professional services vendor the main opportunity is in using these tools and methods to re-engineer applications for their clients. Over the next five years this could well begin to contribute more than 50% of their application development revenues.

There is no doubt that, given the right tools, re-engineering will offer IS departments the best payback on their huge applications investments.

Appendix A
Analysis of User Research Sample

EXHIBIT A

User Survey Sample Analysis

Country	Interviews
France	20
Germany	20
U.K.	20
Number of IS Staff	**Interviews**
500-3000	11
100-500	29
< 100	20
Sector	**Interviews**
Discrete Manufacturing	12
Process Manufacturing	9
Banking & Finance	11
Insurance	6
Others	22
TOTAL INTERVIEWS	60

Appendix B

EXHIBIT B-1

CASE
Market Forecast Database Scenarios

Europe	1991	1992	1993	1994	1995	1996	1997	1992-1997 CAGR (Percent)
France	300	390	585	815	1100	1440	1780	35
Germany	300	345	435	560	725	940	1200	28
United Kingdom	300	320	370	420	500	550	630	15

EXHIBIT B-2

CASE Products
Market Forecast Database by Major Country

Country	1991	1992	1993	1994	1995	1996	1997	1992-1997 CAGR (Percent)
France	90	105	130	160	210	280	350	27
Germany	65	80	105	145	200	280	385	37
United Kingdom	105	115	145	190	235	275	320	23
Rest of Europe	40	45	55	65	80	105	145	26
TOTAL	300	345	435	560	725	940	1200	28